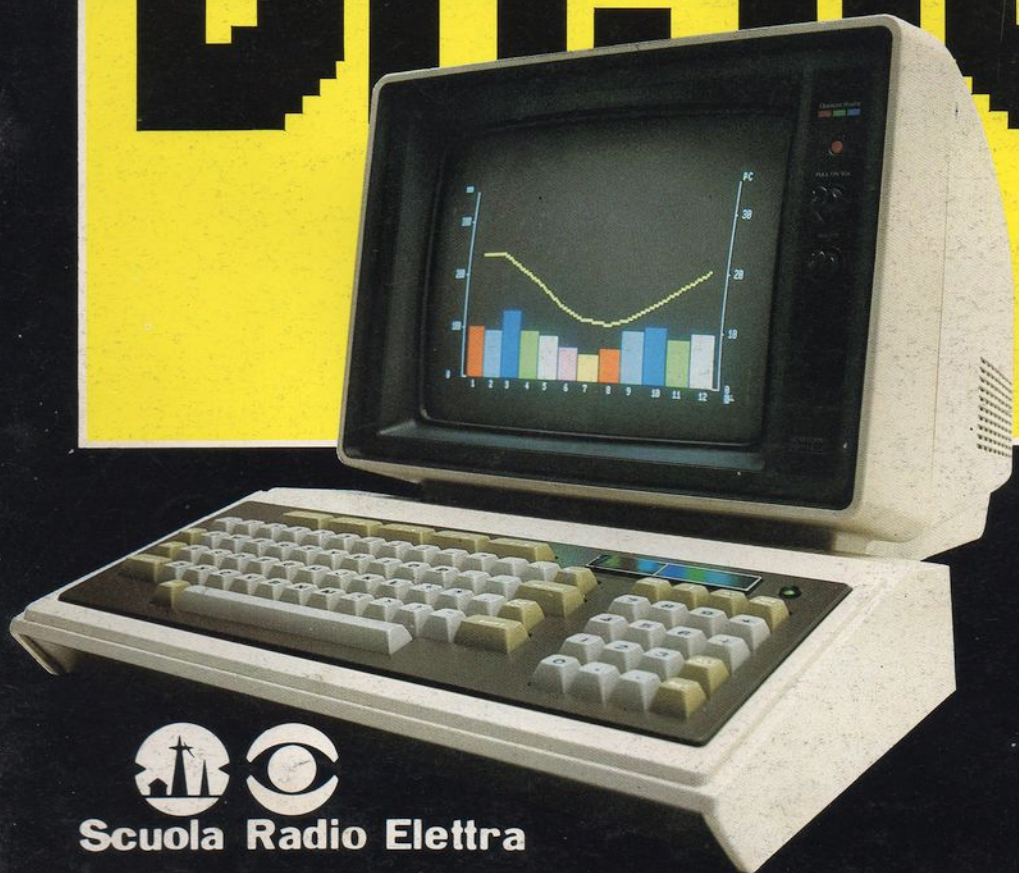
PARLA BASIC



Scuola Radio Elettra

Scuola Radio Elettra
Torino

Analisi e programmazione

Lezione 6

1 — Le istruzioni di diramazione
2 — Le strutture di controllo
3 — Vettori, matrici e tabelle
4 — Gli indici dei vettori e delle matrici

Che cosa apprenderà

In questa lezione sarà completato lo studio delle istruzioni di controllo del programma, prendendo in considerazione l'enunciato **ON... GOTO**.

Quindi si procederà ad analizzare il significato ed i risvolti applicativi dell'istruzione **GOSUB**, che consente di eseguire subroutine (ossia sottoprogrammi) le quali vengono utilizzate per scrivere separatamente parti di programma che devono essere ripetute piú volte.

Verranno, inoltre, esaminate le variabili con indice (vettori e matrici), che consentono di trattare con un unico nome insiemi di variabili omogenee (liste o tabelle di dati).

L'uso dei vettori e delle matrici permette di semplificare molti aspetti della programmazione.

1. – Le istruzioni di diramazione

Nelle precedenti lezioni è stata esaminata l'istruzione di diramazione semplice **IF... THEN....**

Questa istruzione si definisce semplice perché consente di proseguire l'elaborazione per una certa parte del programma (parte indicata dal numero d'istruzione posto dopo la parola THEN...) se la relazione algebrica o logica scritta dopo la parola IF risulta VERA o, in alternativa, di proseguire l'elaborazione per un'altra parte del programma (quella che inizia dall'istruzione successiva all'istruzione **IF**).

Le alternative multiple

In questa lezione analizzeremo l'**alternativa multipla**, che consente all'elaborazione di proseguire, non soltanto per due parti diverse di programma, ma per **piú di due** parti diverse.

L'istruzione BASIC che realizza l'alternativa multipla è:

ON... GOTO...

Dopo la parola riservata **ON** è riportato il nome di una variabile intera (variabile che può cioè assumere solamente valori interi, i quali normalmente partono da 1 ed arrivano fino a 4 o 5).

Dopo la parola **GOTO** c'è una lista di numeri di istruzioni, normalmente quattro numeri come massimo, separati da una virgola. Ad esempio:

```
10 ON I GOTO 50, 80, 40, 500.
```

L'operatività di questa istruzione è la seguente: se la variabile di nome I, al

momento dell'esecuzione dell'istruzione 1Ø, ha il valore 1, l'elaborazione proseguirà per l'istruzione 5Ø; se ha il valore 2, essa proseguirà per l'istruzione 8Ø; se ha il valore 3, proseguirà per l'istruzione 4Ø; infine, se ha il valore 4, salterà all'istruzione 5ØØ.

L'istruzione ON... GOTO...

In altre parole, l'istruzione **ON... GOTO...** si può cosí commentare.

In base (ON) al valore che assume la variabile I, l'elaborazione prosegue per l'istruzione il cui numero di riferimento è indicato nella lista di quattro numeri riportata dopo la parola GOTO.

Con questa istruzione, come si può constatare, si può "alternativamente" saltare in piú di due parti diverse del programma.

A seconda che il linguaggio BASIC sia piú o meno evoluto, nella lista riportata dopo la GOTO possono esserci anche piú di quattro numeri di istruzioni, cioè di alternative.

Nei microsistemi, invece, dove il BASIC ha capacità piuttosto limitate, l'alternativa multipla consente soltanto tre o quattro diverse diramazioni, o addirittura non è nemmeno contemplata nel linguaggio.

2. – Le strutture di controllo

Riassumendo quanto è stato detto circa le istruzioni di diramazione, il linguaggio di programmazione BASIC consente di effettuare deviazioni di elaborazione **sia con condizione sia senza condizione**.

Le istruzioni di diramazione condizionata sono la **IF... THEN...** e la **ON... GOTO...**, mentre le istruzioni di diramazione incondizionata sono i cosiddetti "salti" (istruzioni **GOTO...**, **GOSUB...**) in altre parti del programma.

Con le istruzioni sopra citate il flusso di elaborazione subisce una deviazione dalla normale sequenza operativa, sequenza che consiste appunto nel fatto che le istruzioni vengono eseguite secondo l'ordine con cui sono numerate o memorizzate.

Queste diramazioni vengono indicate nel diagramma di flusso con le raffigurazioni grafiche o strutture di controllo riportate nella *fig. 1*, e già citate nelle precedenti lezioni.

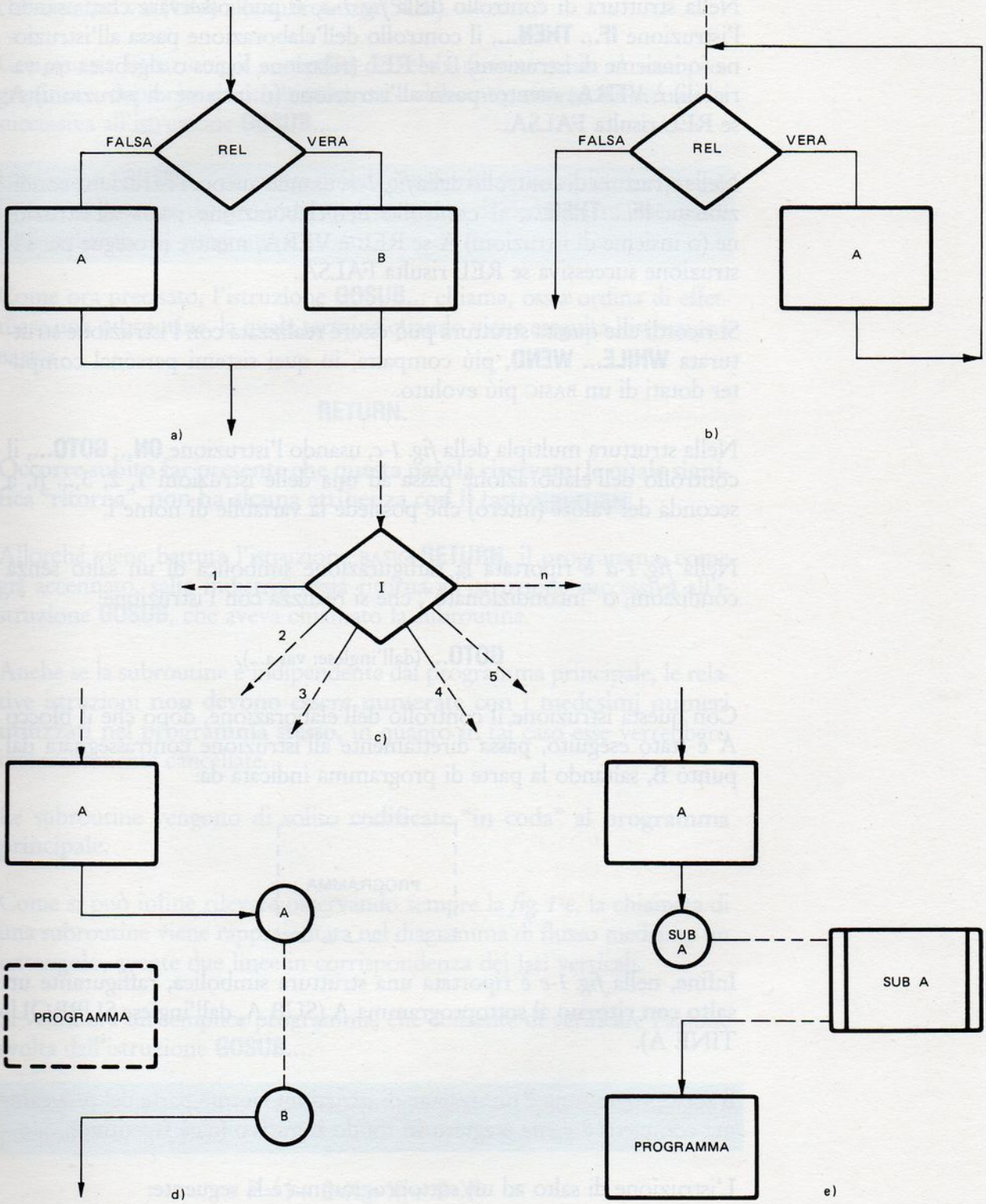


Fig. 1 – Strutture di controllo.

Nella struttura di controllo della *fig. 1-a*, si può osservare che, usando l'istruzione **IF... THEN...**, il controllo dell'elaborazione passa all'istruzione (o insieme di istruzioni) B se REL (relazione logica o algebrica tra variabili) è VERA, mentre passa all'istruzione (o insieme di istruzioni) A se REL risulta FALSA.

Nella struttura di controllo della *fig. 1-b*, usando ancora l'istruzione condizionata **IF... THEN...**, il controllo dell'elaborazione passa all'istruzione (o insieme di istruzioni) A se REL è VERA, mentre prosegue per l'istruzione successiva se REL risulta FALSA.

Si ricordi che questa struttura può essere realizzata con l'istruzione strutturata **WHILE... WEND**, piú compatta, in quei sistemi personal computer dotati di un BASIC piú evoluto.

Nella struttura multipla della *fig. 1-c*, usando l'istruzione **ON... GOTO...**, il controllo dell'elaborazione passa ad una delle istruzioni 1, 2, 3,... n, a seconda del valore (intero) che possiede la variabile di nome I.

Nella *fig. 1-d* è riportata la raffigurazione simbolica di un salto senza condizioni, o "incondizionato", che si realizza con l'istruzione:

GOTO... (dall'inglese: vai a...).

Con questa istruzione il controllo dell'elaborazione, dopo che il blocco A è stato eseguito, passa direttamente all'istruzione contrassegnata dal punto B, saltando la parte di programma indicata da:

Infine, nella *fig. 1-e* è riportata una struttura simbolica, raffigurante un **salto con ritorno** al sottoprogramma A (SUB A, dall'inglese SUBROUTINE A).

Il sottoprogramma è un insieme di istruzioni facente parte del programma completo e viene eseguito **in modo iterativo** (cioè ripetitivo).

L'istruzione di salto ad un sottoprogramma è la seguente:

GOSUB... (numero di programma)

che significa vai (GO) alla subroutine (SUB).

Con questa istruzione il controllo dell'elaborazione passa al sottoprogramma e, dopo che quest'ultimo è stato elaborato, ritorna all'istruzione successiva all'istruzione **GOSUB....**

L'istruzione **GOSUB...** permette quindi di eseguire un completo gruppo di istruzioni separate dal programma principale e di ricorrere ad esse quando è richiesto dal programma stesso.

Come ora precisato, l'istruzione **GOSUB...** chiama, ossia ordina di effettuare una subroutine, la quale termina quando viene eseguita l'istruzione BASIC:

RETURN.

Occorre subito far presente che questa parola riservata, la quale significa "ritorna", non ha alcuna attinenza con il tasto RETURN .

Allorché viene battuta l'istruzione BASIC **RETURN**, il programma, come già accennato, salta indietro, ossia ritorna all'istruzione successiva all'istruzione **GOSUB**, che aveva chiamato la subroutine.

Anche se la subroutine è indipendente dal programma principale, le relative istruzioni **non devono essere numerate con i medesimi numeri utilizzati nel programma stesso,** in quanto in tal caso esse verrebbero inesorabilmente cancellate.

Le subroutine vengono di solito **codificate "in coda" al programma principale.**

Come si può infine rilevare osservando sempre la *fig. 1-e*, la chiamata di una subroutine viene rappresentata nel diagramma di flusso mediante un rettangolo, avente due linee in corrispondenza dei lati verticali.

Si vedrà ora un semplice programma, che consente di verificare l'azione svolta dall'istruzione **GOSUB....**

Si supponga, per esempio, di voler programmare il calcolo del seguente polinomio di 3° grado:

$$P=1+ 2*X +X^2+ 3*X^3$$

per due valori di X (ad esempio, +1 e −1).

NOTA
Viene indicata con il termine "polinomio" la somma algebrica di due o più monomi, ossia espressioni formate da numeri, da lettere, da prodotti di un numero per una lettera, o di un numero per un gruppo di lettere.

Per calcolare le potenze di X, si potrà fare uso di una subroutine (chiamata **SUB** P), la quale riceve i valori di X=−1 e X=+1 ed esegue ogni volta tre potenze:

$P1=X$
$P2=X^2$
$P3=X^3$.

Queste potenze verranno combinate ulteriormente dal programma per eseguire il calcolo finale $P=1+2X+X^2+3X^3$.

Nella *fig. 2* è riportata la codifica in linguaggio BASIC del programma per il calcolo del polinomio in questione.

```
10 REM* CALCOLO POLINOMIO   *
20 REM* P=1+2*X+X↑2+3*X↑3   *
25 REM* PER DUE VALORI DI X *
30 INPUT"INTRODUCI X1 ED X2"; X1,X2
35 LET X=X1
40 GOSUB 100
45 R1=1+2*P1+P2+3*P3
50 LET X=X2
55 GOSUB 100
60 R2=1+2*P1+P2+3*P3
65 PRINT"PER X=";X1,"P(X)=";R1
70 PRINT"PER X=";X2,"P(X)=";R2
75 END
100 P1=X
102 P2=X↑2
104 P3=X↑3
106 RETURN
```

Fig. 2 – Programma BASIC che calcola il polinomio $P=1+2X+X^2+3X^3$.

Come si vede nella *fig. 2*, nel corso del programma il salto al sottoprogramma **SUB** P (che inizia all'istruzione 1ØØ e finisce alla 1Ø6 con l'istruzione **RETURN**) si verifica due volte: una prima volta all'istruzione 4Ø ed una seconda volta all'istruzione 55.

In entrambi i casi l'istruzione **RETURN** della subroutine "ritorna" esattamente all'istruzione successiva a quella da cui era partito il controllo del

programma e cioè la prima volta all'istruzione 45 e la seconda volta all'istruzione 6Ø.

La caratteristica peculiare di una subroutine è di essere utilizzata piú volte dal programma e di ritornare nel punto del programma successivo a quello da cui è stata richiamata (con l'istruzione **GOSUB**).

Nella subroutine si nota altresí che, qualunque sia il valore contenuto nella variabile di nome X, vengono calcolate le potenze:

$$P1=X$$
$$P2=X^2$$
$$P3=X^3.$$

Quindi, la prima volta (quando X=-1) vengono calcolati i valori:

$$P1=-1 \quad (=X)$$
$$P2=+1 \quad (=X^2)$$
$$P3=-1 \quad (=X^3).$$

Tali valori, entrando nella formula del polinomio, fanno assumere a quest'ultimo il risultato:

$$R1=1+2\times(-1)+1+3\times(-1)=$$
$$=1-2+1-3=-3.$$

Analogamente, quando X=1 (cioè la seconda volta) vengono calcolati i valori:

$$P1=1 \quad (=X)$$
$$P2=1 \quad (=X^2)$$
$$P3=1 \quad (=X^3)$$

con cui si otterrà il secondo valore del polinomio:

$$R2=1+2\times(1)+1+3\times(1)=$$
$$=1+2+1+3=7$$

I valori del polinomio cosí calcolati verranno poi visualizzati sullo schermo, per effetto delle istruzioni 65 e 7Ø, nella seguente forma:

```
PER X=-1    P(X)=-3
PER X= 1    P(X)= 7
```

Si può ancora fare notare, osservando il listato della *fig.* 2, che l'istruzione **END** (linea 75) separa il programma principale dalla subroutine.

Nelle prossime lezioni, quando si parlerà delle istruzioni **FOR... NEXT**, si vedrà che il calcolo di un polinomio qualsiasi di grado N si può generalizzare, introducendo come parametri di input sia N sia i coefficienti del polinomio stesso (si ricordi che il grado di un polinomio in una sola variabile X è **la potenza massima di questa variabile**, mentre i coefficienti sono **costanti numeriche** che fungono da moltiplicandi o fattori delle variabili).

A conclusione dell'argomento è bene precisare che la diramazione multipla in un programma può essere anche **con ritorno**, nel senso che il controllo dell'elaborazione passa ad una subroutine e quest'ultima, una volta eseguita, ripassa il controllo all'istruzione successiva a quella di diramazione.

In tal caso non si usa piú l'istruzione:

ON... GOTO...

bensí l'istruzione:

L'istruzione ON... GOSUB...

ON... GOSUB...

RITORNO $\longrightarrow$ ....

Quest'ultima istruzione è molto utilizzata quando nel programma si fa uso di diverse subroutine, per cui la lista di numeri riportata dopo la parola chiave GOSUB si riferisce alle istruzioni iniziali di diverse subroutine.

Approfondiremo in seguito lo studio delle istruzioni **GOSUB** ed **ON... GOSUB....**

3. – Vettori, matrici e tabelle

Nel linguaggio dell'informatica molto spesso compaiono i termini **vettori, matrici** e **tabelle di dati.**

In effetti, queste entità, come anche le costanti e le variabili precedentemente esaminate, **rappresentano insiemi o gruppi organizzati di informazioni**, sui quali si esplica gran parte dell'elaborazione digitale del computer.

I vettori e le matrici, in particolare, sono organizzazioni astratte di dati omogenei (cioè dati o solo numerici o solo alfanumerici), raccolti sotto lo stesso nome.

I vettori e le matrici sono anche denominati genericamente VARIABILI CON INDICI (ARRAY, in inglese).

ARRAY

Cerchiamo ora di comprendere l'utilità pratica che deriva dal trattare un insieme di variabili come un'entità unica.

Si supponga di dover immettere nel computer una serie di variabili numeriche (per esempio, quindici diversi valori) per poterle poi elaborare nei modi previsti dal programma.

In tal caso si devono usare quindici nomi diversi di variabili, ossia uno per ciascun numero, il che crea una notevole complicazione nella stesura del programma; inoltre, qualora si dovesse aggiungere un'altra variabile numerica, occorrerebbe riscrivere il programma per tenere conto del nome della nuova variabile.

In ultimo, si deve tenere presente che nel linguaggio BASIC si possono usare soltanto 286 nomi diversi per contraddistinguere le variabili, per cui è buona norma farne un impiego misurato onde evitare il rischio di non averne piú a disposizione nella stesura di un programma particolarmente lungo.

Per ovviare a questi inconvenienti si ricorre appunto ai vettori ed alle matrici.

Un vettore viene rappresentato utilizzando una lettera od una coppia di caratteri alfanumerici presi dall'alfabeto inglese, seguita da una coppia di parentesi in cui è racchiuso un numero intero (indice). Ad esempio:

A (3) indica il terzo elemento del vettore numerico A;
B (20) indica il ventesimo elemento del vettore numerico B.

In realtà, l'indice non deve essere necessariamente una costante numerica,

bensí può essere costituito da una variabile oppure da una espressione:

```
C (K)
E (L+3)
```

Il vettore viene anche denominato variabile ad un solo indice.

La rappresentazione matematica di un generico vettore è la seguente:

V (J) per J=1, 2, 3, 4,....

Si considerino ora le seguenti due istruzioni:

```
10 A=15
20 B (J)=15.
```

Nell'istruzione 1Ø alla variabile A è sempre assegnato lo stesso valore, ossia 15, mentre nell'istruzione 2Ø il valore 15 può essere assegnato a qualunque elemento del vettore B: il valore della variabile (J) determina quale elemento assume tale valore.

Per esempio, se J=2, il valore 15 è assegnato al secondo elemento del vettore B; se J=7, il valore 15 è assegnato al settimo elemento del vettore B e cosí via.

Un vettore può essere immaginato come una riga o una colonna di valori monodimensionali, ognuno dei quali occupa uno spazio nella memoria come le variabili semplici.

Si supponga di disporre dei seguenti sette valori: 75, 81, 14, 54, 12, 20, 33; questi possono essere immagazzinati in un unico vettore, che chiameremo C, nel seguente modo:

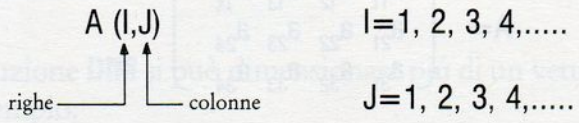
C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)
75	81	14	54	12	20	33

Come si farà ad indicare uno solo di questi sette valori? La cosa è semplice: C (1)=75 sarà il primo elemento della serie, C (2)=81 sarà il secondo elemento e cosí via.

La matrice è invece costituita da una **variabile a due indici** ed è vantaggiosamente utilizzata per rappresentare tabelle di valori omogenei, os-

sia dati disposti in righe e colonne.

Essa viene rappresentata nel seguente modo:



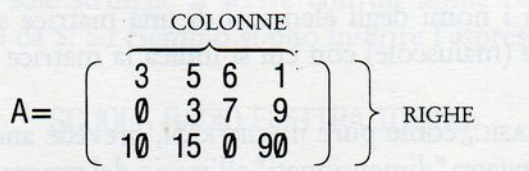
Per le matrici valgono le stesse regole viste per i vettori, tranne che in questo caso dentro le parentesi compaiono due numeri interi (indici), separati da una virgola. Oltre che numeri, entro le parentesi vi possono essere variabili o espressioni.

Per esempio:

C (6,8) indica l'elemento che nella matrice occupa la sesta riga e l'ottava colonna;

D (A+B,7) indica l'elemento che occupa la riga data dal valore (A+B) e la settima colonna.

La rappresentazione matematica di una matrice di nome A, contenente dodici numeri interi pari a tre righe e quattro colonne, è la seguente:



Una matrice di questo tipo è chiamata anche **matrice a due dimensioni.**

Sia i vettori riga sia i vettori colonna vengono denominati matrici ad una sola dimensione.

In genere, il numero delle dimensioni di una matrice è dato dal numero di indici riportati all'interno delle parentesi tonde.

Per esempio, la matrice C (I, J, K, L) è a quattro dimensioni ed ogni dimensione è indicata da uno dei quattro indici I, J, K, L.

In matematica gli elementi di una matrice si indicano genericamente con la stessa lettera con cui si indica la matrice nel suo insieme, ma scritta minuscola.

La matrice A prima considerata si può genericamente indicare nel seguente modo:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

dove ogni elemento (a) di matrice è contraddistinto da due numeri, che precisano rispettivamente **la riga** e **la colonna** di appartenenza dell'elemento in questione.

Ad esempio, a_{11} (si legge "a uno-uno" e non "a undici") indica l'elemento della prima riga e della prima colonna, a_{34} (si legge "a tre-quattro") l'elemento della terza riga e della quarta colonna e cosí via.

Ogni elemento della matrice, come si è visto per il vettore, è costituito da una variabile semplice.

Il numero degli elementi di una matrice si ottiene moltiplicando tra loro le dimensioni massime; nel caso in esame, A contiene dodici elementi **a**, ottenuti come prodotto di 3 (numero massimo della dimensione "riga") per 4 (numero massimo della dimensione "colonna").

In informatica i nomi degli elementi di una matrice sono indicati con le stesse lettere (maiuscole) con cui si indica la matrice nel suo insieme.

Il linguaggio BASIC, come pure il FORTRAN, prevede anche che il vettore e la matrice vengano "dimensionati" all'inizio del programma; questo perché essi possono occupare ampie zone della memoria e quindi occorre che il computer conosca in precedenza quanta memoria deve riservare a ciascun vettore ed a ciascuna matrice.

L'istruzione DIM

Tale dimensionamento consiste nello scrivere un'istruzione (**DIM**), indicando il nome che si vuole assegnare al vettore o alla matrice e le relative dimensioni massime; ad esempio, per dimensionare la matrice A (15,2Ø) basta scrivere:

```
1Ø DIM A (15,2Ø)
```

e per il vettore C (15):

```
1Ø DIM C (15).
```

Se il numero degli elementi di un vettore è inferiore a 10 e se la matrice ha meno di 10 righe e/o 10 colonne, non è necessario indicare nel programma le dimensioni massime di questi vettori e di queste matrici, ossia usare l'istruzione **DIM**.

In un'unica istruzione **DIM** si può dimensionare piú di un vettore o di una matrice. Per esempio:

```
10 DIM A (15,20), B (25), C (15), D (40,20).
```

Poiché i valori degli elementi di una matrice possono essere numeri reali, numeri interi e stringhe, anche i relativi nomi devono essere appropriati.

Occorre cioè usare al piú due caratteri alfanumerici qualsiasi per matrici a valori reali, due caratteri alfanumerici seguiti dal simbolo % per matrici a valori interi e due caratteri alfanumerici seguiti dal simbolo $ per matrici a valori stringa.

Si devono inoltre osservare le limitazioni indicate nella lezione *Analisi e programmazione 5*, a proposito degli intervalli in cui possono essere rappresentati i numeri.

Matrice stringa

Una matrice di sole stringhe si scrive con un nome costituito da due caratteri, seguito da $; ad esempio si può inserire l'espressione:

```
SCUOLA RADIO ELETTRA TORINO
```

in una matrice stringa di 2×2=4 elementi, come precisato di seguito

$$B\$= \begin{pmatrix} \text{"SCUOLA"} & \text{"RADIO"} \\ \text{"ELETTRA"} & \text{"TORINO"} \end{pmatrix}$$

dove:

```
B$ (1,1)="SCUOLA"
B$ (1,2)="RADIO"
B$ (2,1)="ELETTRA"
B$ (2,2)="TORINO"
```

Se poi si volesse preparare un breve programma che visualizzi sullo scher-

mo la scrittura:

SCUOLA RADIO ELETTRA

TORINO

si potrebbe scriverlo cosí:

```
⋮
10 PRINT B$(1,1), B$(1,2), B$(2,1)
20 PRINT "  ↑", B$(2,2)
⋮
```

2 spazi

Con la **PRINT** 1Ø vengono visualizzati gli elementi della matrice stringa B$ (1,1), B$ (1,2), B$ (2,1) su una stessa riga dello schermo, distanziati da un numero fisso di spazi (numero che, come è noto, dipende dal sistema personal computer).

Con la **PRINT** 2Ø vengono visualizzati sulla successiva riga dello schermo una serie di due spazi e, piú distanziata, la stringa B$ (2,2) (che è la parola "TORINO").

Naturalmente, già prima che questi elementi siano stampati o visualizzati, è necessario eseguire una serie di assegnazioni dei valori stringa citati sopra ai corrispondenti nomi degli elementi della matrice B$.

I vettori e le matrici sono entità matematiche, che servono per eseguire svariati calcoli scientifici, detti appunto CALCOLI MATRICIALI; successivamente si studieranno in modo piú approfondito queste entità.

Una tabella è invece **un'organizzazione di dati**, caratterizzata da una uniforme disomogeneità dei vari elementi.

Ogni dato, infatti, è composto da diversi tipi di elementi, ognuno con ricorrenza uniforme.

Esempi di tabelle sono gli elenchi telefonici, il prezzario dei prodotti di un'azienda, l'indirizzario dei clienti.

A titolo esemplificativo nella *fig. 3* è riportata una tabella contenente un elenco di clienti, ordinati per codice-cliente ascendente.

Come si può notare, ogni elemento della tabella è costituito da **due tipi**

di informazioni: da un codice e da una descrizione del cliente (quest'ultima è chiamata anche "valore della tabella").

TABELLA	
CODICE	**CLIENTE**
10010 A	DITTA A
10011 A	DITTA B
10012 A	DITTA C
10013 A	DITTA D
10014 A	DITTA E
10015 B	.
10016 B	.
10017 B	.
10018 B	.

Fig. 3 - Tabella di codici-clienti.

Il codice è ordinato per valori alfanumerici crescenti e funge tra l'altro da **chiave di ricerca**, nel senso che il reperimento programmato di un elemento della tabella viene effettuato eseguendo una ricerca per codici crescenti.

Per esempio, per ricercare la DITTA D, il cui codice è 10013A, si scorre la tabella, partendo dall'inizio ed effettuando passo a passo un confronto logico tra l'elemento oggetto di ricerca (DITTA D) e ciascun elemento della tabella.

Una tabella è quindi caratterizzata dal fatto che ogni elemento è costituito da diverse informazioni, delle quali una assume la funzione di "chiave" o "indice" di ricerca.

Non necessariamente una tabella è ordinata per chiavi crescenti o decrescenti; quando non lo è, per il reperimento di un elemento vengono utilizzati criteri di ricerca diversi da quello visto prima.

Ogni elemento di una tabella si può considerare una stringa (cioè un insieme di caratteri alfanumerici).

L'archiviazione o memorizzazione di una tabella nella memoria princi-

pale si può effettuare ricorrendo ad un'organizzazione vettoriale: ogni elemento (codice+valore) viene assegnato ad un elemento di un vettore stringa.

Con simile organizzazione interna la tabella viene memorizzata secondo una sequenza di dati, ordinata come un vettore.

E' questo il criterio piú comunemente adottato per memorizzare una tabella di dati nei personal computer.

Nei grossi elaboratori esistono diversi e più efficienti tipi di memorizzazione, come ad esempio la "catena", il "plesso", ecc., i quali consentono il reperimento dei dati in un tempo molto breve.

Piú avanti si parlerà di "organizzazione di massa" dei dati; con questo termine si intende il complesso delle strutture fisiche o logiche con cui i dati informativi sono memorizzati nelle cosiddette memorie ausiliarie di massa, quali l'unità a disco e l'unità a nastro.

Ciò che per ora preme sottolineare è che una struttura organizzativa, come quella che si è vista per i vettori, le matrici e le tabelle, è funzionalmente la piú adatta per le elaborazioni matematiche (calcolo matriciale) e logiche (ordinamento alfanumerico).

L'elemento di controllo di questa struttura è sempre costituito da uno o piú INDICI, che collegati al nome consentano al sistema di effettuare la ricerca del "valore" dell'elemento.

L'indice di una tabella o vettore, come già precisato, **è sempre un valore numerico intero**, che può essere una costante o una variabile.

Per esempio, A$(1,4) è un elemento di matrice stringa (di nome A$), individuato dalle coordinate, o coppia di indici, 1 e 4, alle quali corrisponde una ben determinata posizione nella zona di memoria dove è memorizzata la matrice A$.

Anche l'elemento B(I%) è un elemento del vettore di numeri reali B, individuato da un "qualsiasi" valore intero positivo della variabile I%, a cui corrisponde una qualsiasi posizione nell'ambito degli elementi del vettore B memorizzati nella memoria principale del computer.

Un calcolo statistico

Come esempio di quanto detto finora, verrà programmato un calcolo semplice di statistica: la media e la varianza di una serie di numeri casuali, cioè non prevedibili a priori.

Si supponga che i numeri casuali (random, in inglese) siano calcolati dal computer stesso; a tal fine si usa un'istruzione particolare: la **RND** (X).

Questa istruzione, ogni qual volta viene eseguita, **genera un numero a caso, compreso tra il valore Ø ed il valore 1.**

L'istruzione **RND** (X)

L'argomento X può essere un numero qualsiasi: maggiore, uguale o minore di Ø.

In seguito si studierà più nei dettagli questa istruzione, chiamata anche **funzione generatrice di numeri casuali.**

Il programma, quindi, dopo aver generato dei numeri casuali, deve calcolare la media e la varianza di questi numeri.

Matematicamente la media di una serie di numeri si calcola sommando i loro valori e dividendo il risultato per N (numero dei dati); la varianza invece si ottiene calcolando la somma dei quadrati degli scarti della media e dividendo questa somma per N.

Le rispettive formule matematiche sono le seguenti:

$$\mu = \frac{A(1)+A(2)+.....A(N)}{N}$$ (valore medio)

$$\sigma = \frac{(A(1)-\mu)^2+(A(2)-\mu)^2+.....(A(N)-\mu)^2}{N}$$ (varianza)

dove μ e σ sono i simboli, espressi con lettere dell'alfabeto greco, normalmente usati per la media e la varianza di una serie di numeri (nel caso in esame i numeri sono casuali e denominati con i termini di un vettore A (I) per I=1, 2... N, dove N è il numero dei dati casuali).

Nella *fig. 4* è riportato in flow-chart del programma in questione e nella *fig. 5* ne è rappresentata la codifica in linguaggio BASIC.

Nel programma si è supposto che il valore intero di N (numero dei dati casuali) sia 2Ø e che i dati stessi, appena generati, vengano memorizzati in un vettore di nome A (I). Si ha cosí modo di verificare l'uso dei vettori e dell'istruzione **DIM**.

Il programma in oggetto è talmente semplice che si commenta da solo:

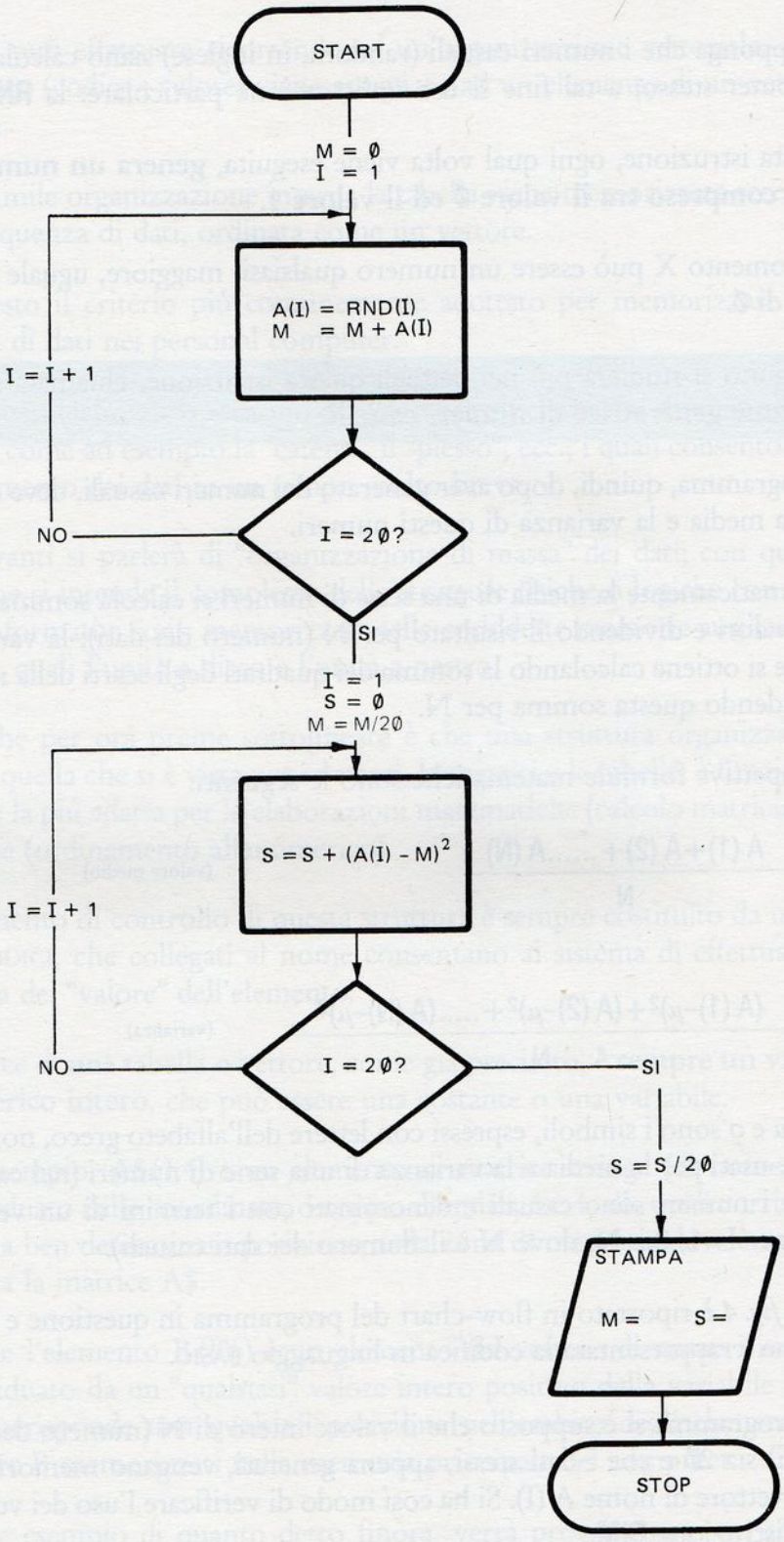


Fig. 4 – Flow-chart del calcolo della media e della varianza di N numeri casuali.

```
10 REM*****************
12 REM*CALCOLO MEDIA *
13 REM*E VARIANZA DI *
14 REM*20 NUMERI CAS-*
15 REM*UALI GENERATI *
16 REM*DAL COMPUTER  *
18 REM*****************
20 DIM A(20)
25 M=0
30 I=1
35 A(I)=RND(I)
40 M=M+A(I)
45 IF I=20 THEN 58
50 I=I+1
55 GOTO 35
58 M=M/20
60 S=0
65 I=1
70 S=S+(A(I)-M)↑2
75 IF I=20 THEN 90
80 I=I+1
85 GOTO 70
90 S=S/20
95 REM*****************
96 REM*ALLA PROSSIMA *
97 REM*ISTRUZIONE    *
98 REM*SONO STAMPATI *
99 REM*SU SCHERMO I  *
100 REM*VALORI M E S *
101 REM*****************
105 PRINT"M=";M,"S=";S
110 END
```

Fig. 5 – Codifica in linguaggio BASIC *del programma della fig. 4.*

esso inizia con l'istruzione 2Ø **DIM** A (2Ø), che definisce il vettore A formato da 20 elementi (numeri dei quali si vuole conoscere la media e la varianza).

Come già accennato, la funzione generatrice **RND** (X) genera un numero a caso, compreso tra Ø e 1, qualunque sia il valore positivo di X: per X si è usata nel programma la variabile intera I, compresa tra 1 e 2Ø e crescente di 1 in 1.

Nelle istruzioni 25 e 3Ø sono posti i valori iniziali di M=Ø (media) e di

I=1; nelle istruzioni 35÷58 viene definito il valore della media dei numeri generati dal computer; nelle istruzioni 6Ø e 65 sono invece posti i valori iniziali di S=Ø (varianza) e di I=1.

Le istruzioni comprese fra 7Ø e 9Ø elaborano i valori di varianza.

Le istruzioni **REM** 1Ø÷18 e 95÷1Ø1 descrivono le fasi di elaborazione che il programma sta per eseguire.

L'istruzione 1Ø5 **PRINT** consente di visualizzare sullo schermo i valori di M e S (media e varianza), preceduti da frasi descrittive (stringhe "M=" e "S=").

Come si vede, tra una stringa ed il risultato c'è un punto e virgola e tra il primo e il secondo risultato c'è una virgola.

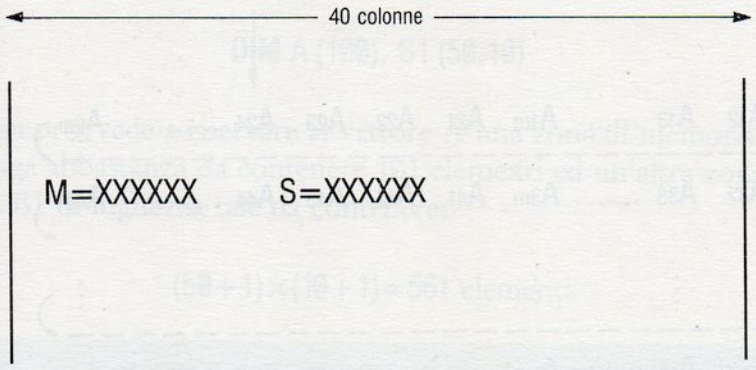
Il primo carattere (;) serve a concatenare due valori qualsiasi ed il secondo (,) a spaziarli di un numero fisso di caratteri.

La **PRINT** al numero 1Ø5, se eseguita su un sistema dotato di schermo a 22 colonne (come il VIC-20), dovrebbe raffigurare i risultati nel seguente modo (naturalmente dopo aver dato il comando di esecuzione del programma **RUN**, seguito da RETURN):

margine sinistro dello schermo
↓

```
M=XXXXXXXXX
S=XXXXXXXXX
```

Se invece la stessa **PRINT** fosse eseguita su un sistema dotato di schermo piú grande (a 40 colonne, per esempio), raffigurerebbe i due risultati sulla stessa riga ma leggermente spostati a sinistra, perché resterebbero liberi i caratteri per un terzo numero.



Ad ogni comando **RUN** appariranno sullo schermo diversi valori di media (M) e varianza (S) dei numeri casuali generati dal computer ad ogni esecuzione del programma.

4. - Gli indici dei vettori e delle matrici

La funzione dell'indice di un vettore

Come già accennato, gli indici dei vettori e delle matrici hanno la funzione di indicare al sistema la posizione dei valori degli elementi nella particolare struttura memorizzata in memoria.

Se, per esempio, si tratta di un vettore e gli elementi di quest'ultimo sono memorizzati uno di seguito all'altro in memoria, il sistema utilizzerà un solo indice per calcolare l'indirizzo di memoria dove è collocato un particolare elemento.

Nel caso invece di una matrice a due dimensioni, verranno utilizzati i due indici di un particolare elemento per calcolare l'indirizzo esatto della memoria in cui esso è collocato.

Ora, se gli elementi di una matrice sono collocati per riga e, per esempio, il primo elemento A (1,1) è memorizzato all'indirizzo I_1, per avere la posizione dell'elemento A (2,4) il sistema eseguirà questo semplice calcolo:

$$I=I_1+(2-1)m+4.$$

Come si vede dalla *fig. 6*, **m** è un numero intero che sta ad indicare il massimo numero delle colonne della matrice A ed **n** è il massimo numero delle righe, sempre della matrice A.

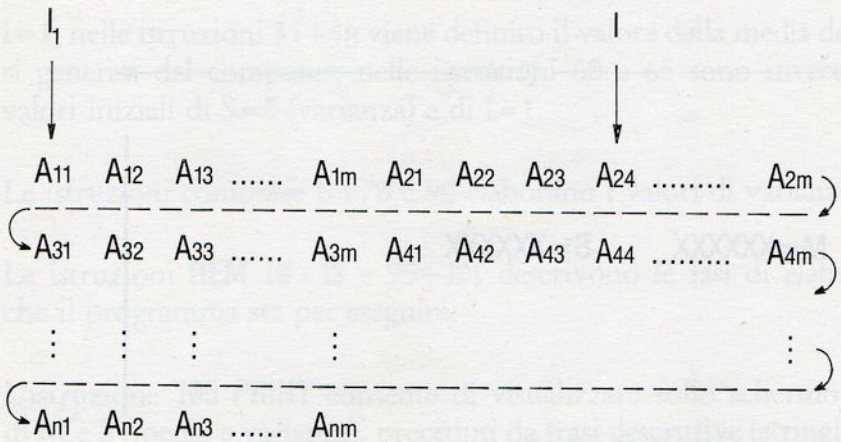


*Fig. 6 – Sistema di memorizzazione in memoria di una matrice di **n** righe e **m** colonne.*

Nella formula precedente i valori 2 e 4 sono proprio gli indici dell'elemento A di cui il sistema cerca la posizione nella zona di memoria dove è memorizzata la matrice, partendo dall'indirizzo I_1 in cui è depositato il primo elemento A (1,1).

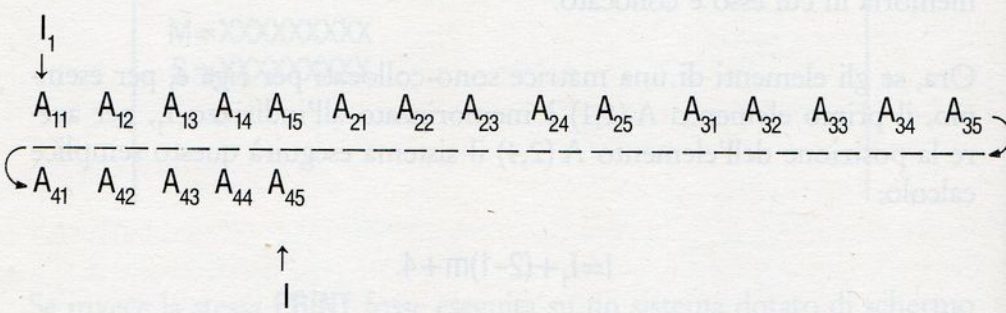
Analogamente, per trovare l'indirizzo dell'elemento A (4,5), il sistema eseguirà il calcolo:

$$I=I_1+(4-1)m+5.$$

Se, per esempio, **m** vale 5 (cioè ci sono 5 colonne), si avrà:

$$I=I_1+3\times5+5=I_1+15+5=I_1+20$$

dove:



L'istruzione DIM per dimensionare una matrice

Come si è visto, le dimensioni massime di un vettore o di una matrice sono specificate nell'istruzione **DIM** all'inizio di un programma.

Scrivendo ad esempio:

DIM A (100), B1 (50,10)

il sistema provvede a riservare al vettore A una zona di memoria principale lunga abbastanza da contenere 101 elementi ed un'altra zona per la matrice B1 di lughezza tale da contenere:

(50+1)×(10+1)=561 elementi.

Il sistema, come si sarà notato, riserva un posto in piú per gli elementi di vettore o matrice (righe e colonne), perché i valori degli indici possono partire da Ø (zero); quindi, ad esempio, da A (Ø) a A (n) ci sono esattamente **n**+1 elementi.

Occorre perciò tenere sempre ben presente che il linguaggio BASIC ammette l'indice Ø per la numerazione degli elementi di un vettore o di una matrice.

Alcuni tipi di computer prevedono l'istruzione BASIC **OPTION BASE** 1, la quale consente di eliminare l'indice Ø e quindi di iniziare i vettori e le matrici dall'elemento 1.

Questo comando deve essere impartito prima che venga usato qualsiasi nome di vettore o matrice.

Come già precisato, l'istruzione **DIM** va anch'essa scritta all'inizio del programma, prima cioè che i vettori o le matrici siano utilizzati nell'elaborazione.

Se ci si dimentica di scrivere un'istruzione **DIM** per vettori e matrici che poi verranno utilizzati nel calcolo, **il sistema provvede autonomamente a riservare loro solo 10 posizioni**, per cui, se dovesse essere utilizzato un elemento A (20) di un vettore A non dimensionato, verrà segnalato "Errore".

Una volta dimensionate, le matrici non possono piú essere "ridimensionate".

L'istruzione ERASE...

Alcuni sistemi personal computer consentono però di ridimensionare (con altre dimensioni) vettori e matrici; in tal caso si devono cancellare dalla memoria del computer, con la particolare istruzione **ERASE...**, i

nomi dei vettori e delle matrici da eliminare, prima di ridimensionare quelli nuovi.

Il numero degli elementi di un vettore o di una matrice, come già visto, si calcola facendo il prodotto delle dimensioni massime; il numero delle dimensioni può essere comunque grande: per esempio, si può dimensionare una matrice di questo tipo:

A (2,2,2,2,2)

costituita da $2^5=32$ elementi di nome A(I,J,K,L,M), con I,J,K,L, M=1,2,3,...

La matrice A in questione viene memorizzata come mostrato nella *fig. 7.*

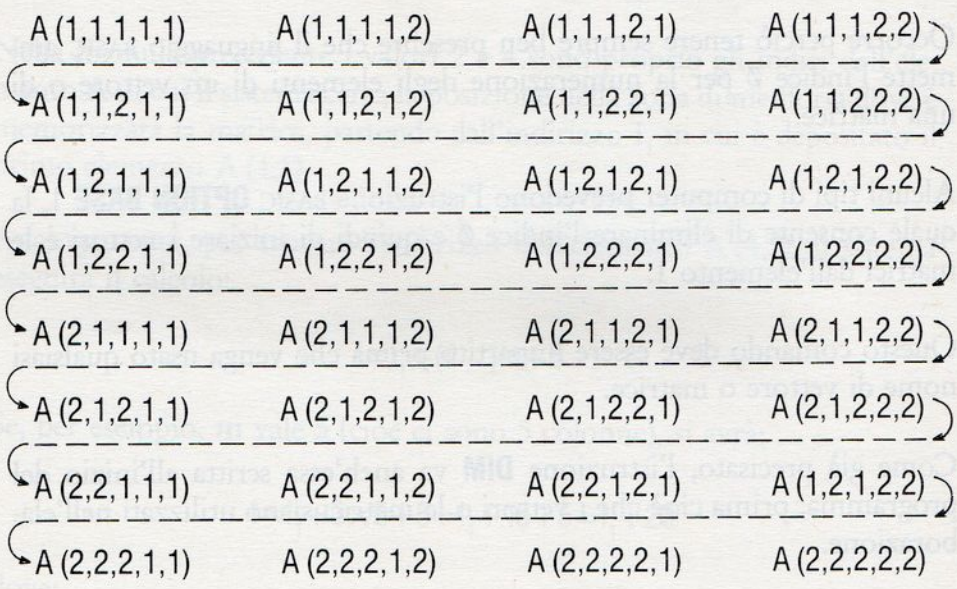


Fig. 7 - Memorizzazione di una matrice A (2,2,2,2,2) a 5 dimensioni, costituita da 32 elementi.

Le dimensioni massime delle matrici dipendono dalle capacità di memorizzazione del sistema personal computer che si sta usando.

* * * * * * *

In seguito...

Nella lezione che seguirà verranno esaminati, fra l'altro, gli enunciati che consentono al programma di ritornare ciclicamente su sé stesso.

Saranno inoltre prese in considerazione le istruzioni DATA, READ e RESTORE, le quali permettono di inserire nel programma delle costanti numeriche o stringa.

Da memorizzare

- L'istruzione **ON...** (A) **GOTO...** (B) realizza una alternativa multipla, dove in (A) è indicata una variabile il cui valore intero consente all'elaborazione di diramarsi per il numero di istruzione specificato in (B).

- SUBROUTINE significa **sottoprogramma**, cioè un complesso di istruzioni che vengono eseguite ripetutamente nel corso dell'elaborazione.

- L'istruzione che permette il salto ad un sottoprogramma è **GOSUB...** (numero di programma).

- Una subroutine termina quando viene eseguita l'istruzione BASIC **RETURN**.

- I **vettori** e le **matrici** sono strutture organizzate di informazioni (numeriche e stringa) **omogenee**, per la cui individuazione nella memoria principale si fa ricorso ad uno o piú indici. Entrambi **sono organizzati sequenzialmente** nella memoria principale e gli indici servono appunto ad individuare la posizione dei vari elementi della struttura.

- I vettori sono particolari matrici aventi una sola dimensione (ed un solo indice).

- Le **tabelle** sono strutture organizzate di informazioni (numeriche e stringa) **disomogenee**, per la cui individuazione si utilizzano sempre uno o piú indici. Esse possono essere organizzate nella memoria principale sequenzialmente come le matrici, ma le singole informazioni vanno rappresentate come stringhe alfanumeriche e successivamente disomogeneizzate in valori numerici e stringa tramite particolari istruzioni.

- Se il vettore o la matrice sono costituiti da piú di 10 elementi, devono essere dimensionati scrivendo all'inizio del programma l'istruzione **DIM...** seguita dal nome del vettore o della matrice riportante tra parentesi le dimensioni massime.

- L'istruzione **RND** (X), ogni qual volta viene eseguita, genera un numero a caso, il cui valore è compreso fra Ø e 1. L'argomento X può essere un numero qualsiasi, maggiore, uguale o minore di Ø.

Scuola Radio Elettra
Torino

Pratica del BASIC

Lezione 6

1 — La cassetta programmi dimostrativi n. 2
2 — La generazione dei numeri casuali
3 — Esercizi di programmazione
4 — I messaggi di errore

Che cosa farà

In questa lezione esaminerà ed eseguirà i due programmi registrati sul lato A della cassetta n. 2, ricevuta con il sesto gruppo di lezioni.

Verificherà poi le azioni svolte dalla funzione **RND**, che consente la generazione dei numeri casuali, e dalla funzione **INT**, che fornisce il valore troncato di un numero decimale.

Entrambe queste funzioni trovano applicazione nei calcoli statistici, nella programmazione di tutti i tipi di giochi di probabilità, ecc.

Avrà modo, quindi, di verificare l'uso delle istruzioni di diramazione e delle funzioni **RND** e **INT** sul computer tramite opportuni esercizi di programmazione.

Completerà la lezione una rassegna dei messaggi d'errore che il computer emette non appena viene eseguita un'istruzione errata.

1. - La cassetta programmi dimostrativi n. 2

Con questo gruppo di lezioni ha ricevuto la seconda cassetta, sul cui lato A sono registrati due distinti programmi dimostrativi in linguaggio BASIC.

Il primo programma è relativo al calcolo dell'ammortamento di un capitale ed il secondo riguarda un interessante gioco matematico, denominato TAXMAN, ossia l'uomo delle tasse.

Prima di caricare questi due programmi nel computer ed eseguirli è opportuno descriverli brevemente.

PROGRAMMA 1: calcolo dell'ammortamento di un capitale

Questo programma, di tipo finanziario, può risultare utile a chi ha interesse a conoscere il piano di ammortamento relativo ad un prestito contratto, ad esempio, presso una banca od un ente assicurativo.

I criteri di ammortamento di un prestito finanziario sono numerosi; tra i piú noti ricordiamo i metodi americano, italiano, francese, tedesco, ecc.

Il metodo italiano, che è stato utilizzato nel programma della cassetta n. 2, consiste nel formulare un piano di rimborso annuale in cui ogni rata è ripartita in due quote: **una quota capitale costante,** che serve ad ammortizzare il prestito o mutuo contratto con l'ente mutuante (banca, assicurazione, ecc.) ed **una quota interessi variabile**, che serve ad ammortizzare il totale degli interessi praticati sul prestito.

La quota interessi decresce con gli anni, perché è calcolata sul debito

residuo che diminuisce di anno in anno: tale diminuzione è costante e vale:

$$-\frac{C}{N}\cdot i\,,$$

dove C è il capitale richiesto in prestito, N è il numero degli anni in cui si vuole ammortizzare il debito contratto, **i** è il tasso di interesse annuale praticato sul debito.

Il metodo italiano si chiama anche **metodo a quote costanti di capitale** e differisce dal metodo francese per il fatto che in quest'ultimo non soltanto le quote interesse variano ma anche le quote capitale (la somma delle quote costituisce una rata annuale costante).

Nel metodo italiano, invece, le rate annuali, costituite da quote capitale costanti e quote interesse variabili, variano e decrescono negli anni.

La quota capitale costante vale:

$$\frac{C}{N}\,.$$

Ad esempio, per ammortizzare con il metodo italiano un debito di L. 10.000.000 in cinque anni, al tasso annuo d'interesse del 20%, si dovrebbero ogni anno versare all'ente mutuante (creditore) rate variabili decrescenti, costituite dalle seguenti due quote.

1) Quota capitale costante del valore di:

$$QC = \frac{C}{N} = \frac{10.000.000}{5} = 2.000.000 \text{ lire}$$

2) Quote interesse variabili, che alla fine del primo anno hanno il valore:

$$QI^1 = C\cdot i = 10.000.000 \times 0{,}20 = 2.000.000 \text{ lire}$$

alla fine del secondo anno diventano:

$$QI^2 = C_i - \frac{C}{N}\cdot i = C\frac{N-1}{N}\cdot i = 10.000.000\cdot\frac{4}{5}\cdot 0{,}20 = 1.600.000$$

alla fine del terzo anno ammontano a:

$$QI^3 = 1.600.000 - 400.000 = 1.200.000 \text{ lire}$$

e così via fino a ridursi, alla fine dell'ultimo anno d'ammortamento, al valore:

$$QI^5 = \frac{Ci}{N} = \frac{10.000.000}{5} \cdot 0{,}20 = 400.000 \text{ lire}$$

Il piano annuale di ammortamento per tale prestito si può quindi raffigurare nel seguente prospetto.

Scadenza	Quota Capitale	Quota Interessi	Rata	Debito Estinto	Debito Residuo
1	2.000.000	2.000.000	4.000.000	2.000.000	8.000.000
2	2.000.000	1.600.000	3.600.000	4.000.000	6.000.000
3	2.000.000	1.200.000	3.200.000	6.000.000	4.000.000
4	2.000.000	800.000	2.800.000	8.000.000	2.000.000
5	2.000.000	400.000	2.400.000	10.000.000	—

Come si può constatare da questo prospetto, le quote capitale sono costanti e ammontano sempre a L. 2.000.000, le quote interesse decrescono in ragione di L. 400.000 all'anno, le rate diminuiscono anche della stessa entità; il debito estinto è la somma delle quote capitale degli anni precedenti fino all'anno corrente, il debito residuo è la differenza tra il debito globale (L. 10.000.000) ed il debito estinto.

Per l'utilizzazione del programma registrato sulla cassetta n. 2 (programma denominato "AMMORT."), occorre inserire i seguenti tre valori.

1) Capitale da ammortizzare
2) Tasso annuo d'interesse
3) Numero anni del piano d'ammortamento.

E' così terminata la descrizione del programma 1; Lei può ora accingersi a caricarlo nella memoria del computer, seguendo la solita procedura.

Inserisca la cassetta nel registratore, **con il lato A rivolto verso l'alto** (il lato B non è registrato) e verifichi che il nastro sia avvolto dall'inizio.

In caso contrario abbassi, il tasto REWIND e, ad avvolgimento avvenuto, prema il tasto STOP.

Batta quindi sulla tastiera del computer l'istruzione di caricamento del programma:

LOAD

seguita da **RETURN** . Prema poi il tasto PLAY del registratore. Dopo un breve intervallo di tempo (2 min. circa), durante il quale il programma viene immesso nella memoria centrale del computer, sullo schermo comparirà la scritta:

READY

Nel caso invece comparisse la scritta ? LOAD ERROR, occorrerà ripetere l'operazione di caricamento del programma, procedendo prima eventualmente ad una pulizia delle testine del registratore mediante un semplice batuffolo di cotone arrotolato attorno ad un bastoncino.

Se nonostante questo accorgimento l'inconveniente dovesse permanere, ripeta l'operazione di pulizia, inumidendo leggermente con alcool il batuffolo di cotone; a questo proposito tenga però presente che, per una buona conservazione delle testine del registratore, **non si deve fare uso frequente dell'alcool** nel corso della loro pulizia.

Dopo aver caricato il programma nella memoria del computer, può eseguirlo battendo il comando BASIC **RUN** seguito da **RETURN** .

Il programma richiederà che siano battuti sulla tastiera i valori:

CAPITALE?
TASSO INT.?
NUM. ANNI?

e visualizzerà sullo schermo, anno per anno, i valori del piano di ammortamento.

Come prova di funzionamento del programma in questione, può per esempio eseguire il calcolo relativo all'ammortamento del prestito di 10.000.000 di lire prima considerato.

Nell'introdurre i valori del piano, tenga presente che, per il capitale chiesto a prestito, deve introdurre un numero reale che rappresenti in lire

italiane il valore da ammortizzare; per il tasso d'interesse deve inserire un valore reale (ad esempio, per il tasso del 15%, batterà 15); per il numero degli anni d'ammortamento deve immettere un valore intero (per esempio, 16, 2Ø, 3Ø, ecc.).

Per rieseguire il programma occorre battere il comando **RUN**.

PROGRAMMA 2: TAXMAN (*l'uomo delle tasse*)

Questo gioco consiste nel prelevare un numero per volta da una serie o successione di numeri (da 5 a 96) da Lei prestabilita all'inizio del gioco stesso.

Ad ogni numero prelevato il computer (l'uomo delle tasse) incamererà i divisori interi del numero stesso: per esempio, se si preleva il numero 12, il computer prenderà i numeri 2, 3, 4, 6 che sono appunto i suoi divisori.

L'abilità del giocatore consiste nello scegliere in modo opportuno i numeri da prelevare, al fine di offrire minori possibilità al computer di incamerare numeri.

A questo scopo è consigliabile iniziare il gioco prelevando il numero primo piú grande presente nella successione di numeri, in quanto è divisibile soltanto per 1.

Durante il gioco potranno essere prelevati unicamente i numeri che hanno divisori; di conseguenza, altri numeri primi presenti nella successione, oppure i numeri che non hanno piú divisori, poiché questi ultimi sono già stati prelevati dal giocatore o dal computer, verranno incamerati dall'uomo delle tasse.

E' consigliabile iniziare il gioco prendendo in considerazione una limitata serie di numeri, per esempio 10 oppure 15 numeri, e poi via via aumentarla sino al numero 96, al fine di rendere il gioco sempre più interessante.

A questo punto è opportuno lasciare a Lei la soddisfazione di scoprire la migliore strategia da adottare per "vincere" il computer.

Per caricare il programma in questione nella memoria del computer deve battere il comando:

LOAD "TAXMAN"

seguito da **RETURN** .

A caricamento avvenuto, può passare ad eseguire il programma scrivendo, come al solito, il comando BASIC:

RUN

2. - La generazione dei numeri casuali

Come già precisato precedentemente, il linguaggio di programmazione BASIC dispone di un'apposita funzione, ossia la **RND** (X), che consente al computer di generare sequenze di numeri decimali casuali, cioè non prevedibili a priori, compresi fra Ø (escluso) e 1 (escluso).

La funzione **RND** trova interessanti applicazioni in molti settori; ad esempio, nella programmazione di tutti i tipi di giochi di probabilità, nel collaudo di programmi, nella simulazione di certe situazioni casuali, nei calcoli statistici, ecc.

In realtà, i numeri forniti dalla funzione **RND** non sono veramente casuali, in quanto sono ottenuti ricorrendo ad una complessa procedura matematica; è quindi piú corretto definirli **pseudocasuali**.

Nella funzione **RND** (X) il valore di X (argomento), che può essere positivo, nullo o negativo, influisce sulla sequenza dei numeri generati. Vediamo ora praticamente come agisce la funzione in questione.

Prima di eseguire gli esercizi previsti, è doveroso precisarLe che, avendo Lei ormai acquisito una certa pratica nell'uso del computer, non verrà piú specificato nel corso delle lezioni che deve premere il tasto **RETURN** al termine di ogni istruzione o comando battuto sulla tastiera.

Accenda il computer ed il televisore.

Batta sulla tastiera la seguente istruzione immediata:

PRINT RND (1)

Sullo schermo apparirà un numero, scritto in notazione decimale, minore di 1: prenda nota di tale numero.

Spenga quindi il computer e, dopo 5 ÷ 6 secondi, lo riaccenda nuovamente.

Batta sulla tastiera la seguente istruzione, che prevede un diverso valore positivo per X:

```
PRINT RND (7)
```

Sullo schermo vedrà comparire lo stesso numero casuale prima visualizzato. Questo è dovuto al fatto che il computer, appena acceso, si trova sempre nella stessa condizione di partenza e non è ancora stato generato nessun numero casuale, per cui, attribuendo a X un qualsiasi valore positivo, sullo schermo comparirà sempre lo stesso numero.

Ripeta la prova ora eseguita, usando però l'istruzione:

```
PRINT RND (Ø)
```

In questo caso, ad ogni accensione del computer, comparirà sullo schermo un numero casuale diverso.

Dopo avere cancellato lo schermo con i tasti SHIFT e CLR | HOME , scriva il seguente, semplicissimo programma che consente di visualizzare la sequenza di numeri casuali generati dal computer:

```
1Ø X=RND (1)
15 PRINT X
2Ø GOTO 1Ø
```

Esegua il programma battendo, come di consueto, il comando **RUN**.

Sul lato sinistro dello schermo verrà visualizzata una serie di numeri, che scorrono con continuità verso l'alto, la maggior parte scritti in notazione decimale ed alcuni in notazione esponenziale.

Per rallentare tale movimento prema il tasto CTRL ; potrà cosí notare che, ad ogni nuovo numero casuale che compare nell'ultima riga inferiore dello schermo, tutti gli altri numeri precedentemente generati vengono spostati verso l'alto.

Dopo questa prova prema il tasto RUN | STOP per fermare il programma. Cancelli quindi ciò che è contenuto nella memoria del computer con il comando BASIC **NEW** ed elimini dallo schermo le scritte con il solito sistema.

Quando il valore di X è negativo, la funzione **RND** dà inizio ad una nuova sequenza di numeri casuali e l'impiego dello stesso numero negativo dà luogo alla medesima sequenza di numeri casuali. In altre parole, per ciascun valore di X si ottiene un numero casuale diverso, ma sempre il medesimo per lo stesso valore di X, come può facilmente verificare scrivendo istruzioni che prevedano un valore negativo per X.

Usando i numeri casuali, si manifesta spesso la necessità di generare numeri compresi fra due valori o limiti.

Ad esempio, per generare una serie di numeri casuali compresi fra Ø e 1Ø, è sufficiente moltiplicare per 1Ø il numero casuale ottenuto per avere il nuovo campo di valori.

Scriva dunque:

PRINT RND (1)*1Ø

Sullo schermo verrà visualizzato un numero compreso fra Ø e 1Ø.

Per ottenere invece un campo di numeri casuali compresi fra due prefissati valori, ad esempio 1Ø e 2Ø, occorre moltiplicare per 1Ø il numero casuale e sommare ad esso 1Ø.

Scriva l'istruzione che segue:

PRINT RND (1)*1Ø+1Ø

Sullo schermo comparirà, come previsto, un numero casuale compreso fra 1Ø e 2Ø.

Molte volte nella generazione dei numeri casuali interessa ottenere numeri interi, ossia privi della parte decimale.

Nel linguaggio BASIC esiste un'apposita funzione, e precisamente la **INT** (X), che fornisce il valore troncato di X, cioè privo delle cifre decimali poste alla destra del punto decimale.

Per verificare quanto asserito, scriva la seguente istruzione:

PRINT INT (23.45)

sullo schermo comparirà, come previsto, il numero:

```
23
```

da cui è stata omessa la parte decimale (.45).

Scriva ora:

```
PRINT INT (-23.45)
```

sullo schermo vedrà apparire:

```
-24
```

In questo caso, essendo negativo il numero da troncare, esso è stato approssimato all'intero immediatamente inferiore al suo valore corrente (tenga presente che -24 è minore di -23).

Il comportamento ora rilevato della funzione **INT** (X) nei confronti dei numeri negativi non è uguale per tutti i computer; in taluni calcolatori, infatti, viene semplicemente omessa la parte decimale, come si è visto per i numeri positivi.

La funzione **INT** viene usata non solamente per troncare un numero, ma anche per arrotondarlo per eccesso o per difetto.

Nel caso in cui la parte decimale del numero X da arrotondare sia eguale o superiore a Ø.5 (.5), l'arrotondamento viene eseguito per eccesso, usando la formula:

```
INT (X+.5)
```

ossia l'arrotondamento viene fatto sommando .5 al numero (X) e poi troncando il numero ottenuto.

Per verificare quanto asserito, scriva la seguente istruzione:

```
PRINT INT (23.55+.5)
```

sullo schermo apparirà il numero:

```
24
```

Come previsto, l'arrotondamento è stato fatto al numero immediatamente successivo.

Se invece la parte decimale del numero è inferiore a .5, esso viene arrotondato al numero intero immediatamente inferiore, utilizzando la formula:

```
INT (X-.5)
```

Per esempio, scriva:

```
PRINT INT (23.45-.5)
```

sullo schermo comparirà il numero 22.

Con l'impiego della funzione **INT**, l'istruzione che consente di ottenere la generazione dei numeri casuali interi in un certo campo assume pertanto la forma:

```
X=INT (RND (1)*A)+B
```

dove A rappresenta il numero dei possibili risultati e B è il numero minore nel campo considerato.

Per ottenere ad esempio un campo di numeri interi compresi fra 1Ø e 2Ø, occorre scrivere la seguente istruzione:

```
PRINT INT (RND (1)*11+1Ø)
```

sullo schermo apparirà un numero casuale intero, compreso fra 1Ø e 2Ø.

Come si può notare, in questo caso il numero casuale è stato moltiplicato per 11, anziché per 1Ø come nell'esempio prima considerato, in quanto la funzione **INT**, troncando le cifre decimali, limita il campo superiore a 19 anziché a 2Ø.

Volendo fare un altro esempio, possiamo considerare la seguente istruzione, che simula il lancio di un dado:

```
X=INT (RND (1)*6+1)
```

In questo caso il numero casuale generato, compreso fra Ø e 1, viene moltiplicato per 6, al fine di espandere il campo da Ø a 6 (in realtà, maggiore di zero e minore di 6); quindi viene aggiunto 1, in modo che il

risultato sia minore di 7.

La funzione **INT** provvede, infine, ad eliminare tutte le cifre decimali, per cui si ottiene un numero compreso fra 1 e 6.

Per verificare quanto specificato, scriva ed esegua il seguente, brevissimo programma:

```
10 X=INT (RND (1)*6+1)
15 PRINT X
```

Ogni volta che verrà battuto il comando BASIC **RUN**, sullo schermo comparirà un numero compreso fra 1 e 6.

3. - Esercizi di programmazione

Come già constatato, l'istruzione grazie alla quale si può realizzare una diramazione semplice (rappresentata da una delle due strutture di controllo della *fig. 1-a* e della *fig. 1-b* descritte all'inizio della lezione *Analisi e Programmazione 6*), è l'istruzione condizionale:

IF... THEN...

Per la diramazione multipla (quattro vie) viene utilizzata l'istruzione condizionale:

ON... GOTO n. 1, n. 2, n. 3, n. 4.

Quest'ultima, come è noto, ha il seguente significato: in base al valore che assume la variabile scritta dopo la parola chiave ON... (valore che può essere costituito da uno qualsiasi dei numeri 1, 2, 3, 4), il controllo dell'elaborazione passa alternativamente ad una delle istruzioni indicate dai numeri di riferimento n. 1, n. 2, n. 3, n. 4.

Se tale variabile vale 1, il controllo passa all'istruzione n. 1; se vale 2, il controllo passa all'istruzione n. 2, e così via; se il suo valore è diverso dai numeri 1, 2, 3, 4, l'elaborazione prosegue per l'istruzione successiva alla **ON... GOTO** n. 1, n. 2, n. 3, n. 4.

Possiamo ora verificare, tramite opportuni esercizi di programmazione,

quanto precisato sull'uso delle due istruzioni di diramazione.

ESERCIZIO 1 – Ricerca del numero minore fra N numeri.

Come certamente ricorderà, nella lezione *Analisi e Programmazione 3* era stato proposto un analogo problema: riveda pertanto il relativo flow-chart e rilegga le spiegazioni fornite al riguardo.

In quel problema si era supposto che la serie di dati numerici A (1), A (2), A (3),... A (N) fosse già presente, memorizzata nella memoria interna del computer; adesso, per contro, supponiamo che questi dati vengano immessi uno per volta dall'unità di input, cioè dalla tastiera.

Nella *fig. 1* è stato disegnato il nuovo flow-chart, che tiene conto di tale modifica. Nella *fig. 2* è invece riportata una possibile codifica in linguaggio BASIC del diagramma della *fig. 1*.

Esaminiamo ora brevemente le fasi operative del programma in questione ed il relativo diagramma di flusso.

Come noterà, il parametro N, numero dei dati della successione numerica, va introdotto all'inizio del programma (istruzione 3Ø).

Inoltre, i dati della successione numerica, denominati A (I), sono dimensionati nel vettore di nome A, denominato A (N), cioè al vettore è riservata tanta memoria quanti sono i dati che si introducono (istruzione 35).

Anche in questo caso, come già visto nella lezione *Analisi e Programmazione 3*, viene indicato con I un contatore ad indice che parte dal valore iniziale 1 e di 1 in 1 arriva al valore N–1 (numero dei dati meno 1).

Dopo l'istruzione **INPUT** "INTRODUCI DATO", il programma si chiede se il valore I=N: in caso affermativo l'elaborazione prosegue tramite le tre istruzioni di assegnazione I=1, DM=A (I), IM=I, dove con DM viene indicato il dato minore e con IM la sua posizione di memoria.

Se invece I non è uguale a N, il programma ritorna all'istruzione 45 (**GOTO** 45).

Nel diagramma di flusso della *fig. 1* quanto ora visto corrisponde al primo simbolo di diramazione semplice.

Dopo le tre istruzioni di assegnazione (65, 7Ø, 75), il controllo dell'elabo-

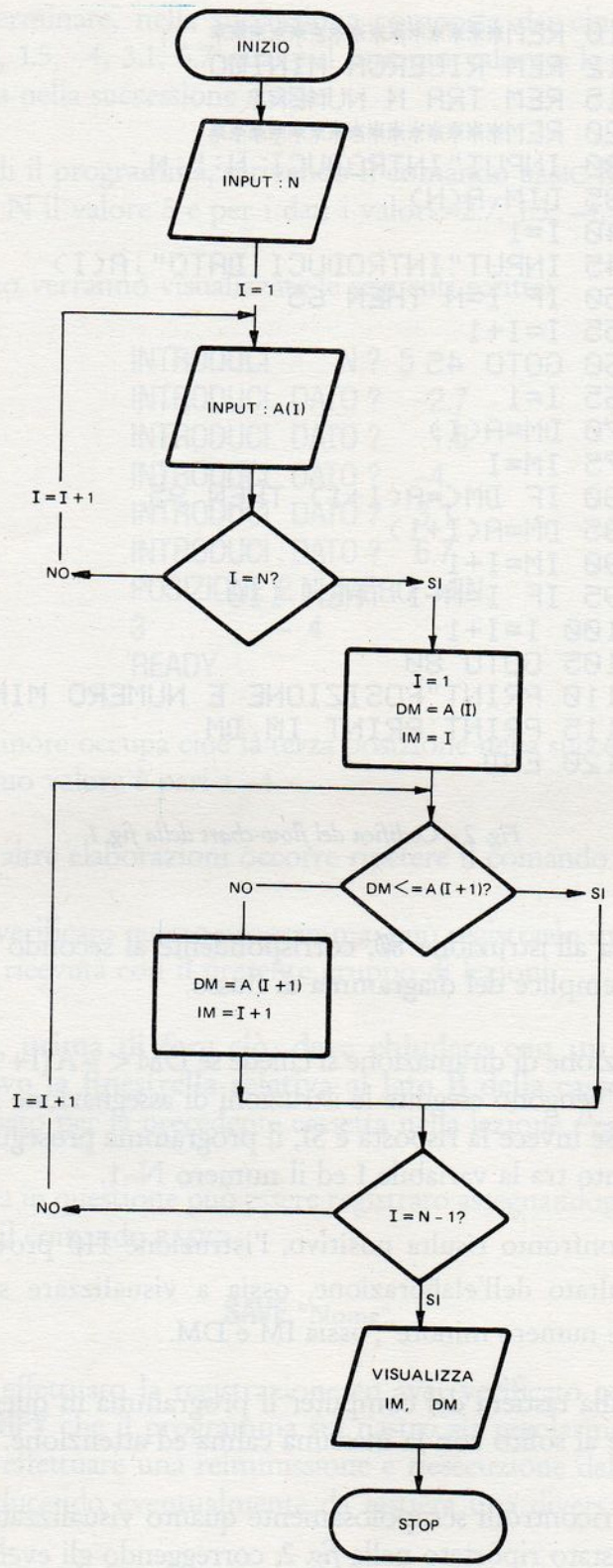


Fig. 1 - Flow-chart del programma che ricerca il numero minore tra N numeri.

```
10 REM***************
12 REM RICERCA MINIMO
15 REM TRA N NUMERI
20 REM***************
30 INPUT"INTRODUCI N:";N
35 DIM A(N)
40 I=1
45 INPUT"INTRODUCI DATO";A(I)
50 IF I=N THEN 65
55 I=I+1
60 GOTO 45
65 I=1
70 DM=A(I)
75 IM=I
80 IF DM<=A(I+1) THEN 95
85 DM=A(I+1)
90 IM=I+1
95 IF I=N-1 THEN 110
100 I=I+1
105 GOTO 80
110 PRINT"POSIZIONE E NUMERO MIN"
115 PRINT:PRINT IM,DM
120 END
```

Fig. 2 – Codifica del flow-chart della fig. 1.

razione passa all'istruzione 8Ø, corrispondente al secondo blocco di diramazione semplice del diagramma di flusso.

Questa istruzione di diramazione si chiede se DM< =A(I+1). Se la risposta è "NO", vengono eseguite le istruzioni di assegnazione DM=A(I+1) e IM=I+1; se invece la risposta è SI, il programma prosegue ad effettuare il confronto tra la variabile I ed il numero N–1.

Se questo confronto risulta positivo, l'istruzione 11Ø provvede a stampare il risultato dell'elaborazione, ossia a visualizzare sullo schermo "posizione e numero minore", ossia IM e DM.

Batta ora sulla tastiera del computer il programma in questione, procedendo come al solito con la massima calma ed attenzione.

Al termine ricontrolli scrupolosamente quanto visualizzato sullo schermo con il listato riportato nella *fig. 2*, correggendo gli eventuali errori.

Per collaudare il programma ora immesso nel computer, possiamo per

esempio determinare, nella successione composta dai cinque seguenti numeri: –2.7, 1.5, –4, 3.1, 6.7 qual è il minimo valore e la posizione da esso occupata nella successione stessa.

Esegua quindi il programma, scrivendo il comando BASIC **RUN** ed introducendo per N il valore 5 e per i dati i valori –2.7, 1.5, –4, 3.1, 6.7.

Sullo schermo verranno visualizzate le seguenti scritte:

```
INTRODUCI      N:? 5
INTRODUCI  DATO ?  -2.7
INTRODUCI  DATO ?   1.5
INTRODUCI  DATO ?  -4
INTRODUCI  DATO ?  3.1
INTRODUCI  DATO ?  6.7
POSIZIONE E NUMERO  MIN.
3            - 4
READY
```

Il numero minore occupa cioè la terza posizione nella successione considerata ed il suo valore è pari a –4.

Per eseguire altre elaborazioni occorre ripetere il comando **RUN**.

Dopo avere verificato questo programma, può registrarlo sul lato B della cassetta n. 2, ricevuta con il presente gruppo di lezioni.

Ricordi che, prima di fare ciò, deve chiudere con un pezzetto di nastro adesivo la finestrella relativa al lato B della cassetta n. 2, come già precisato per la precedente cassetta nella lezione *Pratica 2*.

Il programma in questione può essere registrato assegnandogli un nome a piacere, con il comando BASIC:

SAVE "Nome"

Dopo avere effettuato la registrazione ed aver verificato con il comando BASIC **VERIFY** che il programma sul nastro sia regolarmente registrato, provi ad effettuare una reimmissione e riesecuzione del programma stesso, introducendo eventualmente da tastiera una diversa serie di valori numerici.

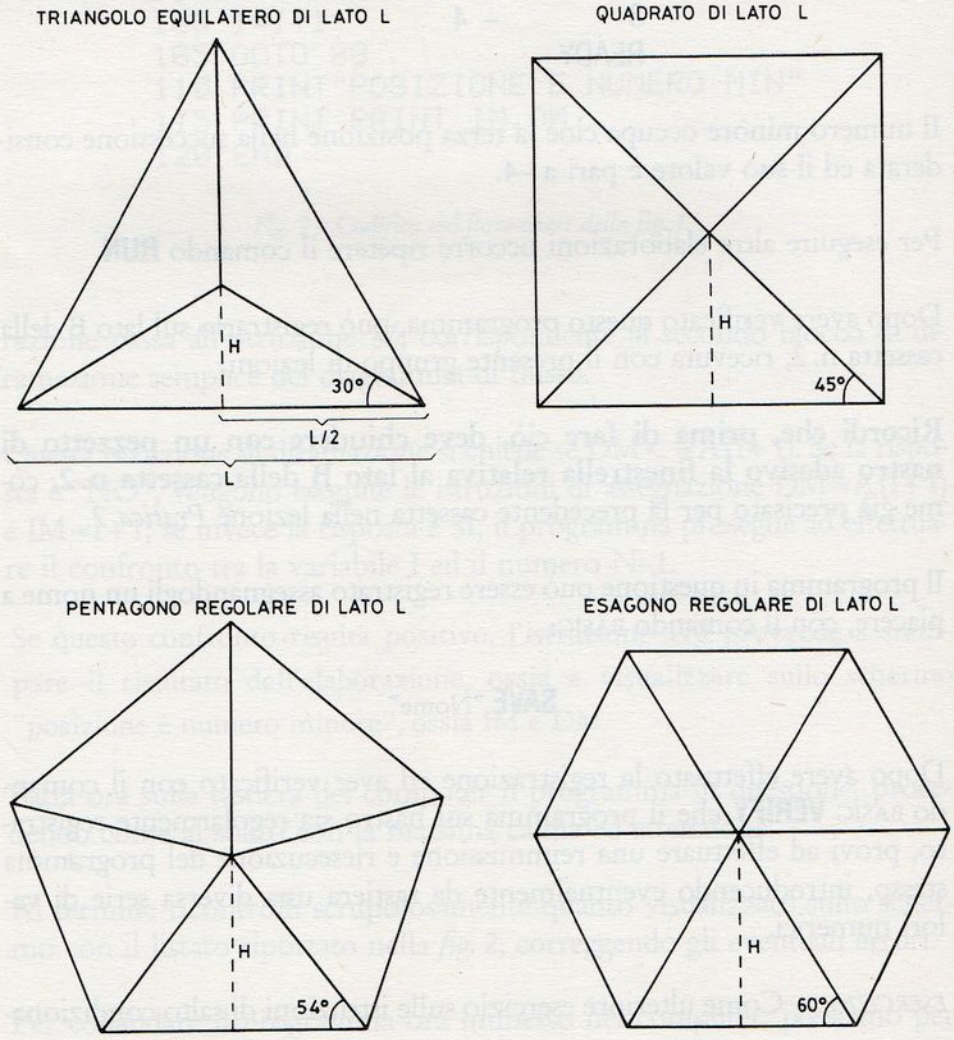
ESERCIZIO 2 – Come ulteriore esercizio sulle istruzioni di salto condiziona-

to, possiamo programmare il problema che calcola il perimetro e l'area di alcuni poligoni regolari (triangoli equilateri, quadrati, pentagoni regolari, esagoni regolari), assegnati la misura del lato (L) ed il numero dei lati (N).

Dalla geometria euclidea sappiamo che un poligono di N lati si definisce regolare se ha tutti gli N lati e gli N angoli interni eguali.

Come è noto, il calcolo del perimetro di un poligono regolare si esegue moltiplicando semplicemente la lunghezza del lato L per il numero N dei lati.

Fig. 3 – Poligoni regolari.



La dimostrazione matematica che segue non è indispensabile per la comprensione del programma; essa viene riportata unicamente per chiarire il procedimento adottato al fine di determinare i fattori F utilizzati nel programma stesso per il calcolo dei quattro poligoni regolari.

Per il calcolo dell'area occorre invece fare alcune osservazioni di carattere geometrico: se si analizza la *fig. 3*, in cui sono riportate le rappresentazioni geometriche dei primi quattro poligoni regolari, si può constatare che l'area di ognuna di queste figure geometriche si può ottenere con la somma di tanti triangoli elementari (triangoli isosceli per il triangolo, il quadrato ed il pentagono, e triangoli equilateri per l'esagono) quanti sono i lati del poligono, cioè N.

Le altezze H dei vari triangoli elementari si ottengono moltiplicando la misura L/2 per la tangente trigonometrica dell'angolo alla base, che varia in relazione al tipo di poligono regolare (30° o $\pi/6$ per il triangolo equilatero, 45° o $\pi/4$ per il quadrato, 54° o $3\pi/10$ per il pentagono, 60° o $\pi/3$ per l'esagono).

Quindi, l'area del triangolo isoscele elementare interno al triangolo equilatero grande si otterrà moltiplicando la misura della base L per l'altezza H, ottenuta come descritto sopra, e dividendo il risultato per 2, cioè:

$$A_T = \frac{L \cdot H}{2} = \frac{L \cdot \frac{L}{2} \cdot \operatorname{tg} \frac{\pi}{6}}{2} = \frac{L^2}{2} \cdot \frac{\operatorname{tg} \frac{\pi}{6}}{2}.$$

Analogamente per il quadrato, l'area del triangolo elementare sarà:

$$A_Q = \frac{L \cdot H}{2} = \frac{L \cdot \frac{L}{2} \cdot \operatorname{tg} \frac{\pi}{4}}{2} = \frac{L^2}{2} \cdot \frac{\operatorname{tg} \frac{\pi}{4}}{2}.$$

Per il pentagono si avrà:

$$A_P = \frac{L \cdot H}{2} = \frac{L \cdot \frac{L}{2} \cdot \operatorname{tg} \frac{3\pi}{10}}{2} = \frac{L^2}{2} \cdot \frac{\operatorname{tg} \frac{3\pi}{10}}{2}.$$

Infine, per l'esagono si otterrà:

$$A_E = \frac{L \cdot H}{2} = \frac{L \cdot \frac{L}{2} \cdot \operatorname{tg} \frac{\pi}{3}}{2} = \frac{L^2}{2} \cdot \frac{\operatorname{tg} \frac{\pi}{3}}{2}.$$

Con il computer possiamo calcolare a parte i fattori non comuni delle precedenti espressioni (cioè i fattori F che contengono le tangenti trigo-

nometriche), eseguendo le seguenti istruzioni e tenendo presente che il simbolo π è ottenuto premendo prima il tasto **SHIFT** e poi il tasto ↑.

a) **PRINT** Ø.5*TAN (π/6)

che visualizzerà il fattore F_T per il triangolo equilatero:

.288675135 (si prenderanno in considerazione solo le prime tre cifre decimali)

b) **PRINT** Ø.5*TAN (π/4)

che visualizzerà il fattore F_Q per il quadrato:

.5

c) **PRINT** Ø.5*TAN (3*π/1Ø)

che visualizzerà il fattore F_P per il pentagono:

.68819Ø96 (si prenderanno in considerazione solo le prime tre cifre decimali)

d) **PRINT** Ø.5*TAN (π/3)

che visualizzerà il fattore F_E per l'esagono:

.866Ø25ØØ3 (si prenderanno in considerazione solo le prime tre cifre decimali)

La superficie S di ciascun poligono si ottiene moltiplicando l'area del triangolo elementare, calcolata prima, per il numero N dei lati e cioè:

$$S_T = 3 \cdot \frac{L^2}{2} \cdot F_T$$ per il triangolo equilatero

$$S_Q = 4 \cdot \frac{L^2}{2} \cdot F_Q$$ per il quadrato

$$S_P = 5 \cdot \frac{L^2}{2} \cdot F_P$$ per il pentagono regolare

$$S_E = 6 \cdot \frac{L^2}{2} \cdot F_E$$ per l'esagono regolare

Si tratta ora di elaborare un programma che preveda l'entrata di due pa-

rametri:

1) N, numero dei lati uguali del poligono regolare di cui si vuole calcolare il perimetro e l'area;

2) L, lunghezza del lato comune del poligono regolare di cui si vuole calcolare il perimetro e l'area;

e che successivamente calcoli e stampi il perimetro e l'area del poligono di lato L, con numero di lati N.

In questo programma verranno assegnati alla variabile di nome F i fattori (o coefficienti) calcolati prima (fattori che avranno il valore corrispondente al tipo di poligono che si sta elaborando).

Nella *fig. 4* e nella *fig. 5* sono riportati rispettivamente il flow-chart e la codifica in BASIC di tale programma, di cui esamineremo ora le fasi operative.

Il programma inizia con l'istruzione **INPUT** N, L, dove N è il numero dei lati del poligono di cui si vuole calcolare il perimetro e l'area, e L è la lunghezza del lato.

In seguito il programma provvede a controllare, mediante le istruzioni 4Ø, 45 e 55, se il numero N è intero, e se è compreso fra 3 e 6.

Questa verifica è rappresentata nel diagramma di flusso del programma (*fig. 4*) mediante i simboli di diramazione condizionale, contraddistinti con le scritte N=INT(N), N>=3, N<=6.

Se una delle suddette condizioni non è verificata, l'elaborazione si ferma (STOP).

Se invece N è compreso fra 3 e 6, il programma prosegue con l'istruzione 7Ø (**ON... GOTO**), che consente di effettuare salti multipli condizionati dal tipo di poligono che si deve elaborare (individuato dal valore N).

In altre parole, questa istruzione permette di scegliere la diramazione del programma corrispondente al tipo di poligono di cui si vuole calcolare il perimetro e l'area.

La fase in cui viene presa questa decisione è rappresentata nel diagramma di flusso dal simbolo di decisione contraddistinto con M.

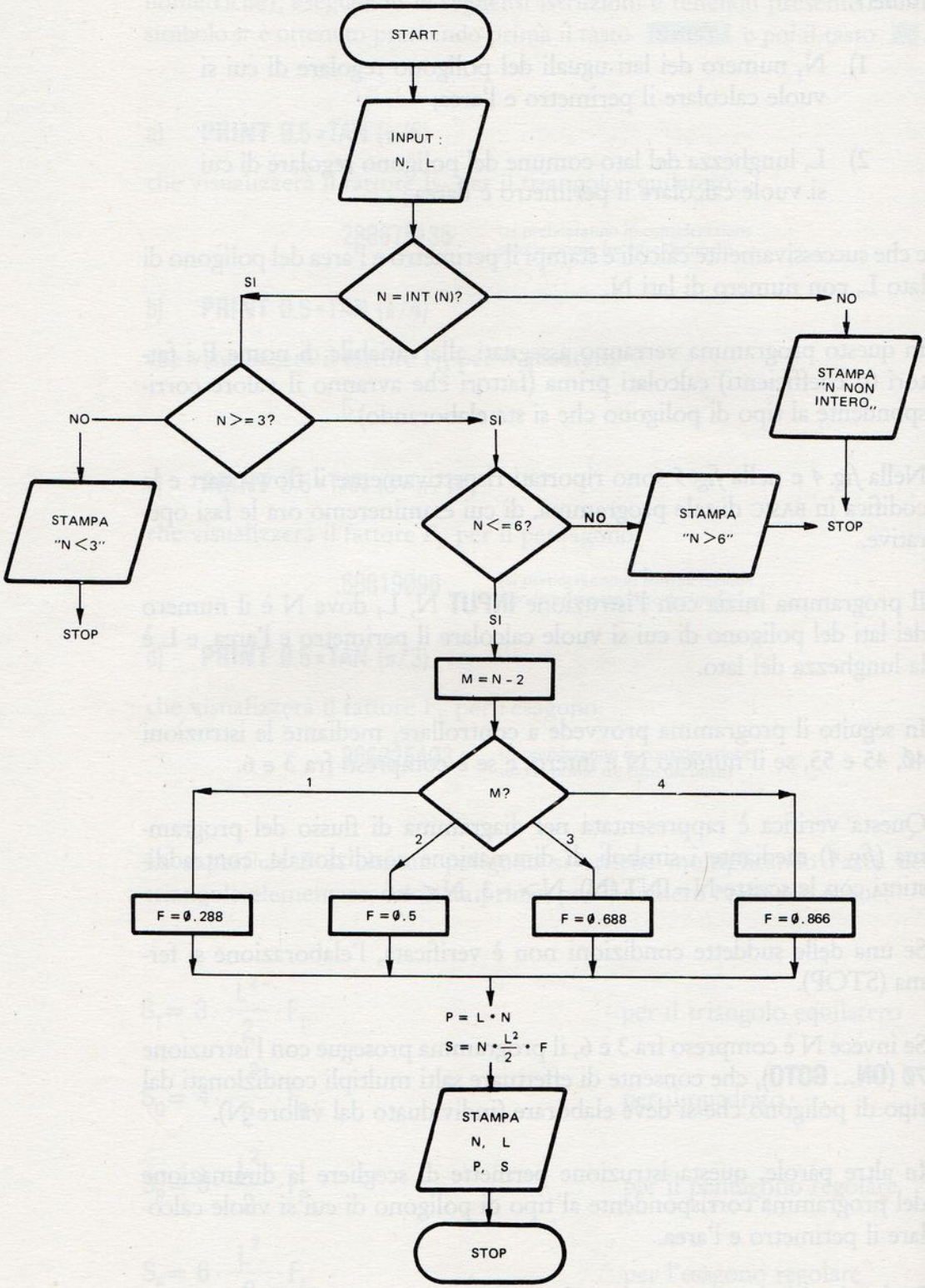


Fig. 4 – Flow-chart del programma che calcola il perimetro e l'area dei primi quattro poligoni regolari.

```
10 REM*****************
15 REM*CALCOLO PERI- *
20 REM*METRO ED AREA *
25 REM*POLIGONI REG. *
30 REM*****************
35 INPUT"INTR.N,L";N,L
40 IF N<>INT(N) THEN 120
45 IF N>=3 THEN 55
50 PRINT"N<3":GOTO 135
55 IF N<=6 THEN 65
60 PRINT"N>6":GOTO 135
65 M=N-2
70 ON M GOTO 75,80,85,90
75 F=0.288:GOTO 95
80 F=0.5:GOTO 95
85 F=0.688:GOTO 95
90 F=0.866
95 P=L*N
100 S=N*(L↑2)/2*F
105 PRINT"N=";N:PRINT"L=";L
110 PRINT"PERIMETRO=";P
115 PRINT"AREA=";S:GOTO 135
120 PRINT"N NON INTERO":GOTO 135
135 STOP
140 GOTO 35:END
```

Fig. 5 – Codifica in BASIC *del programma relativo al calcolo dei perimetri e delle aree di alcuni poligoni regolari.*

Una volta eseguita l'istruzione di assegnazione 75, oppure 8Ø, o 85, o 9Ø, a seconda del tipo di poligono, l'elaborazione prosegue mediante le istruzioni 95 e 1ØØ, le quali provvedono ad eseguire il calcolo del perimetro P e dell'area S.

Dopo di ciò viene visualizzato sullo schermo il risultato ed il programma si ferma con l'istruzione **STOP** alla linea 135, per cui sullo schermo comparirà il messaggio:

BREAK IN 135

L'elaborazione può essere ripresa per calcolare il perimetro e l'area di altri poligoni regolari, aventi un numero di lati ovviamente compreso fra 3 e 6, scrivendo il comando BASIC:

CONT

Con questo comando il programma prosegue con l'istruzione successiva alla linea 135, cioè per l'istruzione:

```
140 GOTO 35
```

Osservando il listato della *fig. 5*, si può infine rilevare che sono state scritte frequentemente due istruzioni sulla stessa linea di programma, usando il separatore due punti (per esempio, sulle linee 5Ø, 6Ø, 75, ecc.).

Il linguaggio BASIC consente infatti di scrivere più istruzioni nella stessa linea o riga di programma, purché la lunghezza di tale riga non superi il massimo consentito dal tipo di computer (due righe fisiche per il computer C64 e quattro righe per il C128).

La scrittura di piú istruzioni su una stessa riga, come pure l'uso di nomi brevi per le variabili, consentono di risparmiare memoria nei programmi, il che è particolarmente importante quando si devono scrivere programmi lunghi.

Terminata la descrizione del programma riportato nella *fig. 5*, può immetterlo nel computer battendo sulla tastiera le istruzioni che lo costituiscono.

Verifichi poi attentamente, come al solito, che non vi siano errori, quindi passi ad eseguirlo.

Per compiere questa operazione può, ad esempio, calcolare il perimetro e l'area di un triangolo equilatero di lato 5, battendo il comando BASIC **RUN** ed introducendo per N il numero 3 (triangolo) e per L il valore 5.

Sullo schermo compariranno le scritte:

```
INTR. N, L? 3,5
N=3
L=5
PERIMETRO=15
AREA= 1Ø.8
BREAK IN 135
```

I valori del perimetro e dell'area del triangolo equilatero di lato 5 risultano rispettivamente 15 e 1Ø.8.

Per eseguire gli stessi calcoli relativi ad altri poligoni regolari batta il co-

mando BASIC **CONT**.

Verificata l'efficienza del programma, può registrarlo sul lato B della cassetta n. 2, in coda al programma registrato prima, relativo alla ricerca del numero minore tra N numeri.

Come già fatto piú volte, deve posizionare la testina di lettura/scrittura del registratore subito dopo la fine del programma precedente; per compiere questa operazione, come già Le sarà noto, occorre eseguire il comando:

VERIFY "Nome precedente"

dove con "nome precedente" si intende la denominazione assegnata al primo programma registrato sulla cassetta.

Il sistema risponderà:

VERIFY ERROR

perché il programma presente in memoria è diverso da quello registrato sul nastro.

Posizionata con tale artificio la testina di lettura/scrittura subito dopo il programma già registrato, può procedere alla registrazione del nuovo programma, assegnandogli un nome a piacere, mediante il comando:

SAVE "Nome"

ESERCIZIO 3 - Programma con numeri casuali generati dal computer.

Il programma in questione, molto semplice, ha lo scopo di esemplificare l'uso della funzione **RND** (X) presentando, sotto forma di gioco, degli esercizi di moltiplicazione fra due numeri interi; questi esercizi possono risultare particolarmente utili agli scolari che si trovano in difficoltà nell'apprendere tale tipo di calcolo elementare con l'insegnamento tradizionale.

Tenga presente, a questo riguardo, che il computer può essere vantaggiosamente impiegato per elaborare una grande varietà di programmi, destinati a facilitare l'apprendimento delle varie materie scolastiche.

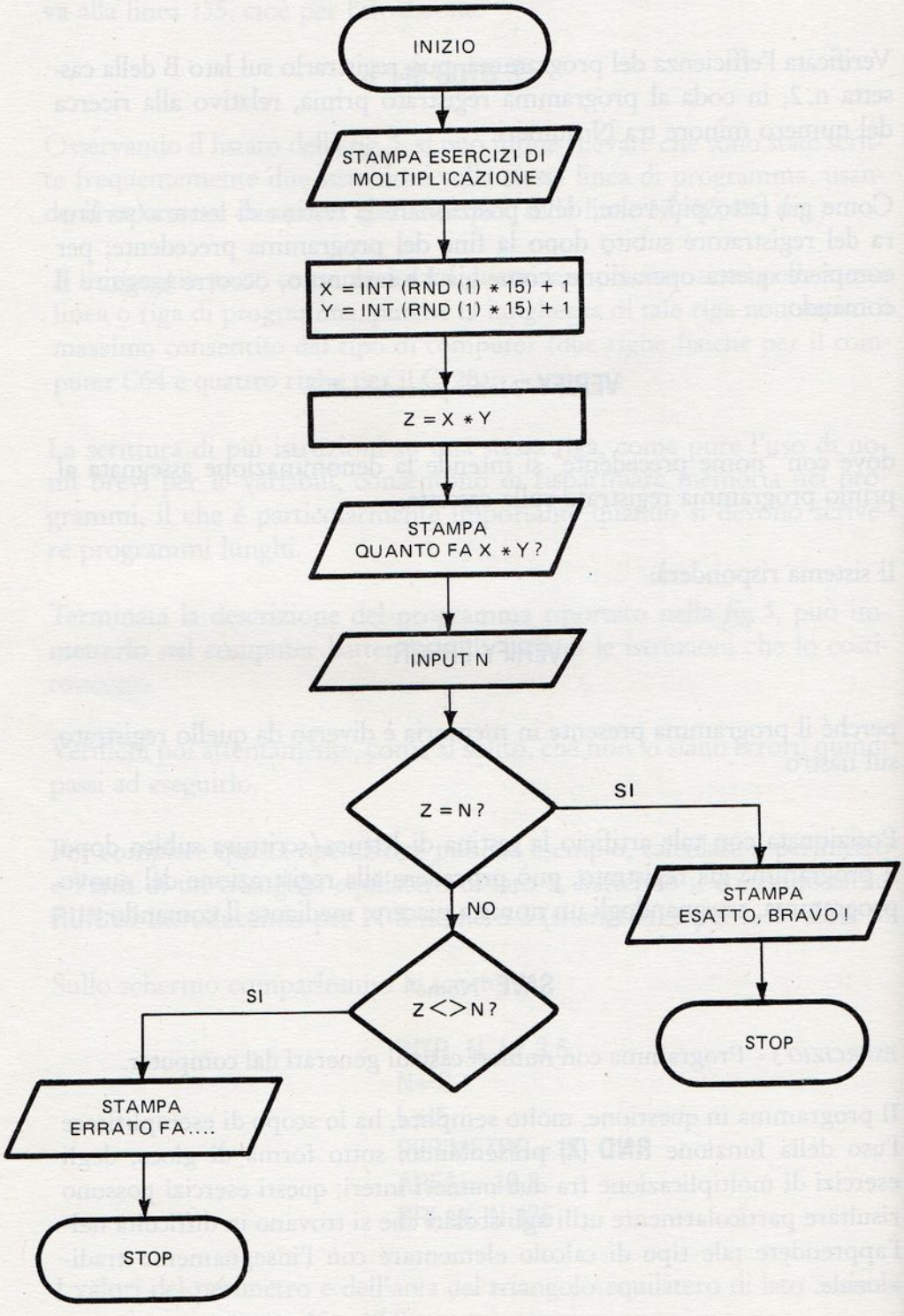


Fig. 6 - Flow-chart del programma dimostrativo della funzione **RND** (X) *generatrice di numeri casuali.*

Nella *fig. 6* e nella *fig. 7* sono rappresentati rispettivamente il flow-chart ed il listato del programma con numeri casuali che ora descriveremo brevemente.

Il programma inizia con l'istruzione **PRINT** che provvede a pulire lo schermo ed a posizionare il cursore nell'angolo superiore sinistro. Per ottenere tale funzione occorre premere, dopo le due virgolette (''), il tasto **SHIFT** e, tenendo questo abbassato, battere il tasto **CLR ‖ HOME**.

Dopo di ciò, una seconda istruzione di stampa visualizza sullo schermo, a partire dalla colonna 7, per effetto della funzione **TAB** (7), la scritta ESERCIZI DI MOLTIPLICAZIONE.

Proseguendo nell'elaborazione, vengono successivamente eseguite le istruzioni (7Ø e 8Ø) X=**INT** (**RND** (1)*15) +1 e Y=**INT** (**RND** (1)*15) +1, che generano numeri interi casuali da 1 (compreso) a 15 (compreso). Tali numeri costituiscono i due fattori della moltiplicazione Z=X*Y, prevista nell'istruzione 9Ø.

All'istruzione 1ØØ il programma visualizza sullo schermo il messaggio che chiede all'operatore "quanto fa" il prodotto dei due fattori X e Y.

A questo punto, mediante l'istruzione 11Ø di **INPUT**, l'operatore deve battere sulla tastiera il numero N corrispondente al risultato della moltiplicazione.

```
10 REM**********************************
20 REM*PROGRAMMA CON NUMERI CASUALI*
30 REM**********************************
40 PRINT"□"
50 PRINT TAB(7)"ESERCIZI DI MOLTIPLICAZIONE"
60 PRINT:PRINT
70 X=INT(RND(1)*15)+1
80 Y=INT(RND(1)*15)+1
90 Z=X*Y
100 PRINT"QUANTO FA";X;"*";Y;
110 INPUT N
120 PRINT:PRINT
130 IF Z=N THEN PRINT"ESATTO BRAVO"
140 IF Z<>N THEN PRINT"ERRATO FA";Z
150 PRINT"----------------------------------------"
160 STOP:GOTO 40
170 END
```

Fig. 7 - Codifica in linguaggio BASIC del programma dimostrativo della funzione **RND** *generatrice di numeri casuali.*

Dopo l'introduzione del numero N, l'elaborazione prosegue con l'istruzione 13Ø, corrispondente al blocco di diramazione semplice del flowchart, contraddistinto con la scritta Z=N?

Questa istruzione si chiede se il valore del numero N battuto sulla tastiera è eguale al valore Z ottenuto dal prodotto dei due numeri casuali X e Y.

Se la risposta è SI, viene stampata la scritta: ESATTO, BRAVO! Se invece la risposta è NO, viene visualizzato il messaggio: ERRATO FA...

Dopo la stampa di uno dei due messaggi citati prima, l'elaborazione si ferma (istruzione 16Ø) e sullo schermo compare la scritta: BREAK IN 16Ø. In questo caso, come già più volte precisato, per eseguire altre moltiplicazioni, occorre battere il comando **CONT**.

La descrizione del listato riportato nella *fig.* 7 è così terminata.

Dopo avere, come al solito, cancellato la memoria e lo schermo, batta sulla tastiera del computer le istruzioni del nuovo programma.

Al termine della battitura verifichi che non vi siano errori, quindi passi ad eseguire il programma con il comando **RUN**.

Sullo schermo comparirà un messaggio che Le chiederà il valore ottenuto dalla moltiplicazione di due numeri casuali compresi fra 1 e 15.

Per esempio, potrebbero presentarsi sullo schermo le seguenti scritte:

ESERCIZI DI MOLTIPLICAZIONE
QUANTO FA 12*13 ?

Battendo sulla tastiera il numero 156, che è appunto il risultato corretto di tale moltiplicazione, sullo schermo comparirà il seguente altro messaggio:

ESATTO, BRAVO!

Se invece in tal caso si battesse una risposta errata, per esempio il numero 158, il computer farà comparire sullo schermo la scritta:

ERRATO FA 156

Per eseguire altre moltiplicazioni, batta il comando **CONT**.

Naturalmente il programma in questione può essere modificato e reso più facile per i bambini che frequentano le prime classi delle scuole elementari, sostituendo il numero 15 in entrambe le funzioni **RND** (istruzioni 7Ø e 8Ø) con un numero minore (ad esempio, 9).

4. - I messaggi di errore

Di seguito viene fornita una prima lista di messaggi d'errore, che il sistema segnala non appena viene eseguita un'istruzione errata. Questa lista si riferisce a tutte le istruzioni finora studiate; altri messaggi verranno elencati e spiegati nelle successive lezioni.

MESSAGGIO D'ERRORE	SIGNIFICATO
BAD SUBSCRIPT	INDICE ERRATO Compare quando l'indice di un elemento di vettore, matrice o tabella è errato, cioè non consente al sistema di ricercare l'elemento.
CAN'T CONTINUE	NON POSSO CONTINUARE Questo messaggio appare quando il comando BASIC **CONT** non funziona dopo l'arresto temporaneo del programma ottenuto con l'istruzione **STOP**, o quando l'elaborazione non può continuare perché è stata corretta una riga di programma oppure quando il programma non è mai stato eseguito (con il comando **RUN**).
DIVISION BY ZERO	DIVISIONE PER ZERO Segnala che è stata eseguita un'istruzione in cui risulta una divisione con denominatore nullo: la divisione per zero non ha senso matematicamente.
EXTRA IGNORED	IGNORATI GLI EXTRA Appare quando sono stati battuti troppi dati in risposta ad un'istruzione **INPUT**.

FILE NOT FOUND

FILE NON TROVATO
Questo messaggio compare allorché un comando BASIC che ricerca un file con un certo nome su nastro o disco non ha esito positivo, cioè il file con quel nome non viene trovato.

FORMULA TOO COMPLEX

FORMULA TROPPO COMPLESSA
Avverte che l'espressione algebrica scritta dopo la parola chiave di un'istruzione è troppo lunga: in tal caso occorre spezzarla in due o più espressioni algebriche.

ILLEGAL DIRECT

ISTRUZIONE IMMEDIATA ILLECITA
Per esempio, l'istruzione **INPUT**, che può essere usata ad esecuzione differita e mai immediatamente.

ILLEGAL QUANTITY

QUANTITA' ILLECITA
Segnala che un argomento di una funzione è fuori del campo ammesso. Ad esempio, l'argomento X negativo della funzione radice **SQR** (X) non è ammesso.

OUT OF MEMORY

FUORI MEMORIA
Compare quando non c'è un'altra memoria interna RAM disponibile per contenere programmi e dati.

OVERFLOW

SUPERO DI CAPACITA'
Indica che un numero introdotto è fuori del campo ammesso: per esempio, il numero 10^{40} (dieci elevato all'esponente 40) è fuori del campo consentito all'elaborazione (il campo va da 10^{-38} a 10^{+38}).

REDIM'D ARRAY

MATRICE RIDIMENSIONATA
Una matrice può essere dimensionata una sola volta. Se viene usata l'istruzione **DIM** per dimensionare una matrice, il sistema automaticamente provvede a riservare a quest'ultima una zona di memoria lunga 10 elementi.

REDO FROM START — RIFARE DALL'INIZIO

Indica che sono stati battuti dati alfanumerici in un'istruzione **INPUT**, mentre si aspettavano dati numerici.
E' sufficiente ribattere i dati in modo corretto.

STRING TOO LONG — STRINGA TROPPO LUNGA

Compare quando è stata battuta una stringa di caratteri alfanumerici superiore a 255 (limite massimo).

SYNTAX — ERRORE DI SINTASSI

Informa che è stato commesso un errore formale di sintassi in un'istruzione.

TYPE MISMATCH — MANCATA CORRISPONDENZA DI TIPO

Questo errore si verifica quando si introduce o viene usato un numero al posto di una stringa o viceversa.

UNDEF'D STATEMENT — ISTRUZIONE INDEFINITA

Segnala che è stato compiuto un salto ad un'istruzione con un numero di riga inesistente.

VERIFY — VERIFICA

Indica che il programma registrato su disco o nastro non corrisponde a quello presente nella memoria principale RAM.

* * * * * * * *

In seguito...

Nella prossima lezione avrà la possibilità, fra l'altro, di verificare sul computer, mediante appositi esercizi di programmazione, i risvolti applicativi delle istruzioni **FOR... NEXT...**, **READ** e **DATA**.

Nelle lezioni che seguiranno potrà trovare riferimenti al computer C64 ed al computer VIC-20. Ovviamente Lei dovrà attenersi a quanto specificato per il computer in Suo possesso, tenendo presente che se esso è il C128 dovrà seguire quanto detto per il C64.

CONTROLLO DIDATTICO 6

Domande

1) - In un programma può essere scritta piú di una subroutine?

.....

.....

2) - Quale errore è contenuto nella seguente subroutine?

```
100 GO SUB 140
 :
140 REM**SOTTOPROGRAMMA**
150 IF A>50 THEN 170
160 A=A+10
170 A=A↑2
180 REM**FINE SOTTOPROGRAMMA**
```

.....

.....

.....

3) - Illustrare gli effetti prodotti dalla seguente istruzione:

```
10 DIM A(11), B(15,10)
```

.....

.....

.....

.....

.....

4) - L'istruzione **DIM** può essere posta in un qualsiasi punto del programma?

.....

.....

.....

.....

5) - Scrivere l'istruzione che consente al computer di generare numeri interi a caso, compresi fra 1 e 1000.

.....

.....

6) - Che cosa verrà visualizzato sullo schermo eseguendo sul computer il seguente, semplice programma?

```
10 J=0
15 J=J+1
20 PRINT J
25 IF J=10 THEN 35
30 GOTO 15
35 END
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

7) - Descrivere brevemente le fasi operative del programma riportato nella domanda n. 6 e disegnare il relativo flow-chart.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

8) - Qual è l'istruzione BASIC, presente in alcuni tipi di computer, che consente di cancellare un vettore oppure una matrice precedentemente dimensionata in un programma?

.....

.....

.....

9) - Individuare l'elemento A(1,2) della seguente matrice:

$$A = \begin{pmatrix} 3 & 8 & -7 & 10 \\ 1.1 & 20 & 12 & 9 \\ 5 & 8 & 1.7 & -5 \end{pmatrix}$$

.....

.....

.....

.....

10) - Che cosa compare sullo schermo eseguendo il seguente brevissimo progamma? Perché?

```
10 X=-5
15 IF X<0 THEN 25
20 PRINT X, "E' NEGATIVO"
30 END
```

.....

.....

.....

.....

.....

11) - Nella *fig. 1* e nella *fig. 2* della *Pratica del* BASIC *6* sono riportati rispettivamente il flow-chart e la codifica di un programma che ricerca il numero minore tra una serie di N numeri.
Se si desidera modificare il programma in modo che ricerchi il numero maggiore tra una serie di numeri, quali varianti si devono apportare ad esso?

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Risposte

1) - Un programma scritto in BASIC può contenere piú di una subroutine.

2) - L'errore risiede nella mancanza, dopo la linea di programma 17Ø, dell'istruzione BASIC **RETURN**, che indica la fine della subroutine.

3) - L'istruzione in oggetto provvede a dimensionare un vettore ed una matrice: il vettore A di 12 elementi e la matrice B di 16 righe e 11 colonne, in quanto nel linguaggio BASIC sono ammessi gli indici zero.

4) - No, l'istruzione **DIM** deve essere posta all'inizio del programma, in modo da precedere tutte le istruzioni in cui vengono utilizzati i vettori e le matrici in essa definiti.

5) - L'istruzione che consente di generare numeri casuali interi, compresi fra 1 e 1ØØØ, è la seguente:

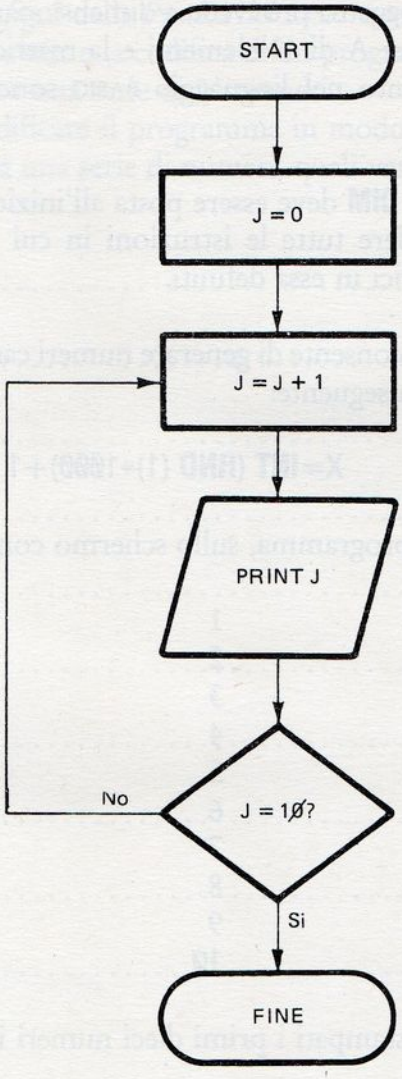
```
X=INT (RND (1)*1ØØØ)+1
```

6) - Eseguendo tale programma, sullo schermo compariranno:

```
1
2
3
4
5
6
7
8
9
1Ø
```

ossia, verranno stampati i primi dieci numeri interi.

7) - Il programma riportato nella domanda n. 6 introduce il concetto di contatore.
L'elaborazione inizia con l'istruzione 1Ø, che assegna il valore Ø alla variabile J. Dopo di ciò il controllo passa all'istruzione 15, dove il precedente valore di J viene incrementato, ossia aumentato, di 1 e quindi stampato con l'istruzione 2Ø.
L'istruzione 25 controlla se il ciclo di calcolo è stato ripetuto per il numero di volte prestabilito e, quando tale valore è raggiunto (J=1Ø), l'elaborazione termina.
L'istruzione 3Ø provvede a fare ritornare il controllo dell'elaborazione all'istruzione 15 dove, come già detto, il valore J viene incrementato di 1 ad ogni esecuzione, sino a quando non raggiunge il valore 1Ø.
Il flow-chart del programma in questione è riportato di seguito.



8) - L'istruzione che consente di cancellare un vettore o una matrice, precedentemente dimensionata in un programma, è la seguente:

ERASE.. (nome di vettore o matrice da cancellare)

9) - L'elemento A (1,2) è relativo alla prima riga e alla seconda colonna, ossia nel caso specifico è il numero 8.

10) - Sullo schermo comparirà il messaggio:

? UNDEF'D STATEMENT ERROR IN 15

perché è stato compiuto un salto ad un'istruzione con un numero di righe (25) inesistente.

11) - Occorre modificare le quattro istruzioni numerate 12, 15, 8Ø e 11Ø; precisamente, occorre scrivere le seguenti nuove istruzioni.

```
12 REM RICERCA MASSIMO
15 REM TRA N NUMERI
:
8Ø IF DM>=A(I+1) THEN 95
:
11Ø PRINT "POSIZIONE E NUMERO MAX"
:
```

* * * * * * *

Pagina mancante

PARLA BASIC

Corso teorico-pratico per corrispondenza di

ANALISI E PROGRAMMAZIONE BASIC

Scuola Radio Elettra

10126 Torino - Via Stellone 5

Tel. (011) 674432